

# **Справочное руководство GDL ArchiCAD 11**

GRAPHISOFT®

## **Graphisoft**

Посетите веб-сайт Graphisoft <http://www.graphisoft.com> для получения дополнительной информации о дистрибьюторах и имеющихся программных продуктах.

## **Справочное руководство GDL ArchiCAD 11**

Авторские права ©2007 Graphisoft, все права защищены. Воспроизведение, изложение и перевод без предварительного получения письменного разрешения строго запрещены.

## **Торговые знаки**

ArchiCAD и ArchiFM являются зарегистрированными торговыми знаками, а PlotMaker, Virtual Building, StairMaker и GDL - торговыми знаками Graphisoft.

Все другие торговые знаки являются собственностью соответствующих владельцев.

---

## **Введение**

*Это пособие является полным справочным руководством по языку GDL (Geometric Description Language - язык геометрических определений), являющемуся собственностью Graphisoft. Это пособие рекомендуется тем пользователям, которые хотят расширить возможности конструктивных инструментов и библиотечных элементов, имеющиеся в ArchiCAD Graphisoft. В нем приводится подробное описание языка GDL, включая описание синтаксиса, команд, переменных и т.п.*



# СОДЕРЖАНИЕ

<b>Содержание</b> .....	<b>5</b>	<b>Простые типы</b> .....	<b>25</b>
<b>Общий обзор</b> .....	<b>13</b>	<b>Производные типы</b> .....	<b>26</b>
<b>Введение</b> .....	<b>13</b>	<b>Преобразование координат</b> .....	<b>27</b>
<b>Написание скриптов</b> .....	<b>13</b>	<b>Преобразования в двумерном пространстве</b> ..	<b>27</b>
Структура библиотечного элемента .....	13	<b>Преобразования в трехмерном пространстве</b> ..	<b>28</b>
Анализ, декомпозиция и упрощение .....	14	<b>Управление стеком преобразований</b> .....	<b>31</b>
Разработка .....	15	<b>Пространственные фигуры</b> .....	<b>33</b>
Простейшие команды .....	15	<b>Основные пространственные фигуры</b> .....	<b>33</b>
Команды средней сложности .....	16	BLOCK .....	33
Сложные команды и дополнительные возможности ..	18	BRICK .....	33
Профессиональное использование GDL .....	19	CYLIND .....	33
<b>Как создаются трехмерные изображения</b> .....	<b>19</b>	SPHERE .....	34
Трехмерное пространство в ArchiCAD .....	20	ELLIPS .....	34
Для чего нужно преобразование координат? .....	20	CONE .....	35
Интерпретатор GDL .....	21	PRISM .....	35
Порядок анализа GDL-скриптов .....	21	PRISM_ .....	36
<b>Синтаксис GDL</b> .....	<b>23</b>	CPRISM_ .....	39
<b>Предложения</b> .....	<b>23</b>	BPRISM_ .....	40
<b>Строки</b> .....	<b>23</b>	FPRISM_ .....	41
<b>Метки</b> .....	<b>23</b>	HPRISM_ .....	43
<b>Символы</b> .....	<b>23</b>	SPRISM_ .....	43
<b>Строки символов</b> .....	<b>24</b>	SLAB .....	46
<b>Идентификаторы</b> .....	<b>24</b>	SLAB_ .....	46
<b>Переменные</b> .....	<b>24</b>	CSLAB_ .....	47
<b>Параметры</b> .....	<b>25</b>	CWALL_ .....	47
		BWALL_ .....	49

XWALL_ .....	51	LIGHT .....	91
XWALL_{2} .....	53	PICTURE .....	93
BEAM .....	55	<b>Объемные текстовые элементы .....</b>	<b>94</b>
CROOF_ .....	55	TEXT .....	94
MESH .....	58	RIGHTTEXT .....	95
ARMC .....	59	<b>Примитивные элементы .....</b>	<b>95</b>
ARME .....	60	VERT .....	96
ELBOW .....	60	TEVE .....	96
<b>Плоские фигуры в трехмерном пространстве 61</b>		VECT .....	96
HOTSPOT .....	61	EDGE .....	97
LIN_ .....	61	PGON .....	97
RECT .....	62	PIPG .....	98
POLY .....	62	COOR .....	98
POLY_ .....	62	BODY .....	100
PLANE .....	63	BASE .....	102
PLANE_ .....	63	<b>Плоскости сечения в 3D .....</b>	<b>102</b>
CIRCLE .....	63	CUTPLANE .....	102
ARC .....	64	CUTPOLY .....	104
<b>Фигуры, создаваемые из ломаных линий .... 64</b>		CUTPOLYA .....	106
EXTRUDE .....	66	CUTSHAPE .....	108
PYRAMID .....	69	CUTFORM .....	108
REVOLVE .....	71	<b>Команды над объемными элементами .....</b>	<b>110</b>
RULED .....	74	GROUP .....	114
RULED{2} .....	75	ENDGROUP .....	114
SWEEP .....	77	ADDGROUP .....	114
TUBE .....	80	SUBGROUP .....	114
TUBEA .....	84	ISECTGROUP .....	114
COONS .....	86	ISECTLINES .....	114
MASS .....	89	PLACEGROUP .....	115
<b>Элементы визуализации .....</b>	<b>91</b>	KILLGROUP .....	115

SWEEPGROUP .....	116	<b>Включение чертежей в смету элементов ...</b>	<b>133</b>
<b>Использование двоичного формата .....</b>	<b>116</b>	DRAWING2 .....	133
<b>Двумерные фигуры .....</b>	<b>119</b>	DRAWING3 .....	133
<b>Чертежные элементы .....</b>	<b>119</b>	DRAWING3{2} .....	134
HOTSPOT2 .....	119	DRAWING3{3} .....	134
LINE2 .....	119	<b>Графическое редактирование .....</b>	<b>135</b>
RECT2 .....	120	<b>Команда редактирования с помощью узловых точек .....</b>	<b>135</b>
POLY2 .....	120	HOTSPOT .....	135
POLY2_ .....	121	HOTLINE2 .....	140
POLY2_A .....	122	HOTARC2 .....	140
POLY2_B .....	122	<b>Коды статусов .....</b>	<b>141</b>
POLY2_B{2} .....	122	<b>Синтаксис кодов статусов .....</b>	<b>141</b>
POLY2_B{3} .....	123	<b>Дополнительные коды статусов .....</b>	<b>143</b>
ARC2 .....	123	Предыдущее состояние ломаной: заданы текущее положение и направляющая .....	143
CIRCLE2 .....	124	Отрезок по абсолютным координатам второй точки ..	143
SPLINE2 .....	124	Отрезок по относительным координатам второй точки .....	144
SPLINE2A .....	126	Отрезок по длине и направлению .....	144
PICTURE2 .....	127	Отрезок по длине вдоль направляющей .....	145
PICTURE2{2} .....	127	Установить начальную точку .....	145
<b>Текстовые элементы .....</b>	<b>127</b>	Замкнуть ломаную .....	146
TEXT2 .....	127	Установить направляющую .....	146
RIGHTTEXT2 .....	128	Установить точку центра .....	147
<b>Использование двумерных данных в двоичном формате .....</b>	<b>128</b>	Дуга, касающаяся направляющей и оканчивающаяся в точке .....	147
FRAGMENT2 .....	128	Дуга, касающаяся направляющей и имеющая заданные радиус и угол .....	148
FRAGMENT2 .....	128	Дуга по точке центра и точке на радиусе конца дуги ..	148
<b>Проекция трехмерных фигур на плоскость .</b>	<b>129</b>		
PROJECT2 .....	129		
PROJECT2{2} .....	129		
PROJECT2{3} .....	132		

Дуга по точке центра и углу	149	DEFINE SOLID_FILL	174
Окружность по ее центру и радиусу	149	DEFINE EMPTY_FILL	174
<b>Реквизиты</b>	<b>153</b>	Определение типа линии	175
<b>Директивы</b>	<b>153</b>	DEFINE LINE_TYPE	175
Директивы, используемые в 3D- и 2D-скриптах	153	DEFINE SYMBOL_LINE	175
[LET]	153	Определение стиля	176
RADIUS	154	DEFINE STYLE	176
RESOL	155	DEFINE STYLE {2}	177
TOLER	156	Определение абзаца	177
PEN	156	Определение текстового блока	178
LINE_PROPERTY	157	Дополнительные данные	179
[SET] STYLE	157	Зависимость от внешнего файла	180
SET STYLE 0	157	<b>Негеометрические скрипты</b>	<b>181</b>
Директивы, используемые только в 3D-скриптах	157	<b>Скрипт спецификаций</b>	<b>181</b>
MODEL	157	DATABASE_SET	181
[SET] MATERIAL	158	DESCRIPTOR	182
SECT_FILL	159	REF DESCRIPTOR	182
SHADOW	160	COMPONENT	182
Директивы, используемые только в 2D-скриптах	161	REF COMPONENT	183
DRAWINDEX	161	BINARYPROP	183
[SET] FILL	161	SURFACE3D ( )	183
[SET] LINE_TYPE	162	VOLUME3D ( )	183
<b>Определение реквизитов</b>	<b>162</b>	POSITION	184
Определение покрытия	163	DRAWING	184
DEFINE MATERIAL	163	<b>Скрипт параметров</b>	<b>185</b>
DEFINE MATERIAL BASED_ON	165	VALUES	185
DEFINE TEXTURE	166	PARAMETERS	186
Определение штриховки	168	LOCK	187
DEFINE FILL	168	HIDEPARAMETER	187
DEFINE FILLA	171	<b>Скрипт интерфейса пользователя</b>	<b>187</b>
DEFINE SYMBOL_FILL	173		



UI_DIALOG .....	187	SQR .....	199
UI_PAGE .....	187	Тригонометрические функции .....	200
UI_CURRENT_PAGE .....	188	ACS .....	200
UI_BUTTON .....	188	ASN .....	200
UI_SEPARATOR .....	189	ATN .....	200
UI_GROUPBOX .....	189	COS .....	200
UI_PICT .....	189	SIN .....	200
UI_STYLE .....	190	TAN .....	200
UI_OUTFIELD .....	190	PI .....	200
UI_INFIELD .....	191	Трансцендентные функции .....	200
UI_INFIELD {2} .....	191	EXP .....	200
UI_INFIELD{3} .....	191	LGT .....	200
UI_TOOLTIP .....	194	LOG .....	201
<b>Выражения и функции</b> .....	<b>195</b>	Логические функции .....	201
<b>Выражения</b> .....	<b>195</b>	NOT .....	201
DIM .....	195	Статистические функции .....	201
VARDIM1(expr) .....	196	MIN .....	201
VARDIM2(expr) .....	196	MAX .....	201
<b>Операторы</b> .....	<b>198</b>	RND .....	201
Арифметические операторы .....	198	Битовые функции .....	201
Операторы отношения .....	198	BITTEST .....	201
Логические операторы .....	198	BITSET .....	201
<b>Функции</b> .....	<b>199</b>	Специальные функции .....	202
Арифметические функции .....	199	Строковые функции .....	202
ABS .....	199	STR .....	202
CEIL .....	199	STR .....	202
INT .....	199	STR{2} .....	202
FRA .....	199	SPLIT .....	205
ROUND_INT .....	199	STW .....	206
SGN .....	199	STRLEN .....	206
		STRSTR .....	206
		STRSUB .....	207

<b>Управляющие предложения</b> _____	<b>209</b>	доступны только в смете и выносных надписях . . . . .	227
<b>Предложения передачи управления</b> . . . . .	<b>209</b>	Параметры объектов и источников света - доступны только в смете и выносных надписях . . . . .	227
FOR . . . . .	209	Параметры дверей, окон и концов стен . . . . .	227
NEXT . . . . .	209	Параметры окон и дверей - доступны только в смете и выносных надписях . . . . .	228
DO . . . . .	210	Параметры источников света - доступны только в смете и выносных надписях . . . . .	229
IF . . . . .	212	Параметры выносных надписей . . . . .	230
GOTO . . . . .	213	Параметры стен - доступны для дверей и окон . . . . .	231
GOSUB . . . . .	213	Параметры стен - доступны только в смете и выносных надписях . . . . .	233
RETURN . . . . .	213	Параметры колонн - доступны только в смете и выносных надписях . . . . .	234
END / EXIT . . . . .	214	Параметры балок - доступны только в смете и выносных надписях . . . . .	236
<b>Манипулирование буфером параметров</b> . . . . .	<b>214</b>	Параметры перекрытий - доступны только в смете и выносных надписях . . . . .	237
<b>Макрообъекты</b> . . . . .	<b>218</b>	Параметры крыш - доступны только в смете и выносных надписях . . . . .	238
<b>Предложение вывода на экран</b> . . . . .	<b>220</b>	Параметры заштрихованных областей - доступны только в смете и выносных надписях . . . . .	239
<b>Операции для работы с файлами</b> . . . . .	<b>220</b>	Параметры 3D-сетки - доступны только в смете и выносных надписях . . . . .	240
OPEN . . . . .	221	Глобальные переменные свободные для использования . . . . .	241
INPUT . . . . .	221	Старые глобальные переменные . . . . .	243
VARTYPE . . . . .	221	<b>Запросы</b> . . . . .	<b>245</b>
OUTPUT . . . . .	221	REQ . . . . .	245
CLOSE . . . . .	222	REQUEST . . . . .	246
<b>Разное</b> _____	<b>223</b>	<b>Двери и окна</b> . . . . .	<b>257</b>
<b>Глобальные переменные</b> . . . . .	<b>223</b>	Общие положения . . . . .	257
Общие параметры рабочей среды . . . . .	223		
Информация об этажах . . . . .	224		
Информация о съемке . . . . .	224		
Общие параметры элементов . . . . .	225		
Параметры объектов, источников света, дверей и окон . . . . .	226		
Параметры объектов и источников света . . . . .	226		
Параметры объектов, источников света, дверей и окон -			

Создание библиотечных элементов дверей и окон . . . . .	258	D . . . . .	281
3D-варианты . . . . .	258	E . . . . .	283
Прямоугольные окна и двери в прямолинейных стенах . . . . .	258	F . . . . .	283
Непрямоугольные двери и окна в прямолинейных стенах . . . . .	260	G . . . . .	284
WALLHOLE . . . . .	260	H . . . . .	284
WALLNICHE . . . . .	264	I . . . . .	284
Прямоугольные двери и окна в криволинейных стенах . . . . .	265	K . . . . .	285
Непрямоугольные двери и окна в криволинейных стенах . . . . .	267	L . . . . .	285
2D-варианты . . . . .	268	M . . . . .	285
Вырезание специального проема в стене . . . . .	268	N . . . . .	286
WALLHOLE2 . . . . .	268	O . . . . .	286
WALLBLOCK2 . . . . .	269	P . . . . .	287
WALLLINE2 . . . . .	269	R . . . . .	289
<b>Скрипт GDL, создаваемый из плана этажа . . . . .</b>	<b>270</b>	S . . . . .	292
<b>Ключевые слова . . . . .</b>	<b>271</b>	T . . . . .	293
Общие ключевые слова . . . . .	271	U . . . . .	294
Устаревшие ключевые слова . . . . .	271	V . . . . .	295
Ключевые слова только для 3D-фигур . . . . .	272	W . . . . .	296
Ключевые слова только для 2D-символов . . . . .	274	X . . . . .	296
Ключевые слова для 2D-символов и 3D-фигур . . . . .	275	Соглашение по именованию параметров . . . . .	298
Ключевые слова негеометрических скриптов . . . . .	275	<b>Расширение GDL Data In/Out . . . . .</b>	<b>298</b>
Скрипт спецификаций . . . . .	275	Описание базы данных . . . . .	298
Скрипт параметров . . . . .	276	Открытие базы данных . . . . .	299
Скрипт интерфейса . . . . .	276	Чтение данных из базы данных . . . . .	300
Алфавитный список ключевых слов GDL . . . . .	277	Запись в базу данных . . . . .	301
A . . . . .	277	Закрытие базы данных . . . . .	301
B . . . . .	277	<b>Расширение GDL DateTime . . . . .</b>	<b>302</b>
C . . . . .	278	Открытие канала . . . . .	302
		Чтение информации . . . . .	303
		Закрытие канала . . . . .	303
		<b>Расширение GDL File Manager I/O . . . . .</b>	<b>304</b>
		Указание папки . . . . .	304

---

Получение имени файла/папки .....	304	CLOSE .....	311
Завершение просмотра папки .....	305	INPUT .....	311
<b>Расширение GDL Text I/O .....</b>	<b>306</b>	OUTPUT .....	313
Открытие файла .....	306	<b>Расширение GDL XML .....</b>	<b>314</b>
Чтение значений .....	307	Открытие документа XML .....	315
Запись значений .....	308	Чтение документа XML .....	316
Закрытие файла .....	309	Изменение документа XML .....	319
<b>Расширение Property GDL .....</b>	<b>310</b>	<b>Предметный указатель .....</b>	<b>323</b>
OPEN .....	310		

---

# ОБЩИЙ ОБЗОР

GDL - это параметрический **язык программирования**, подобный языку BASIC. С его помощью описываются объемные 3D-тела, например, двери, окна, мебель, и 2D-символы, представляющие эти объекты на плане этажа. Эти объекты называются **библиотечными элементами**.

## ВВЕДЕНИЕ

Знание основ программирования, а также начертательной геометрии, несомненно, помогут Вам при освоении GDL.

Не приступайте к практическому освоению GDL с использованием сложных объектов. Изучайте GDL путем последовательного освоения его возможностей. Следуйте рекомендациям согласно уровням квалификации, описанным ниже.

Если Вы знакомы с языком программирования, подобным BASIC, то можете приступить к изучению GDL путем анализа существующих скриптов. Вы также можете многое познать, открыв библиотечные элементы, поставляемые с Вашим программным обеспечением, и познакомившись с GDL 2D- и 3D-скриптами. Кроме того, Вы можете сохранить любой элемент, расположенный на плане этажа, в виде GDL и затем можете познакомиться с его скриптом.

Если Вы не знакомы с языком BASIC, однако имели дело со строительными блоками, Вы все же можете найти свой собственный путь практического освоения GDL. Мы советуем Вам начать написание скриптов из простейших команд и все время проверять в 3D-окне, какие фигуры они строят.

*Для ознакомления с деталями окружающей среды создания и редактирования библиотечных элементов см. раздел "Параметрические объекты" в главе "Виртуальное здание" справки ArchiCAD.*

Graphisoft опубликовал несколько книг по программированию в GDL и созданию библиотечных объектов. Книга "Object Making With ArchiCAD" является прекрасным руководством для начинающих. Книга Дэвида Никольсона "GDL Cookbook" является наиболее популярным курсом по изучению GDL начинающими и квалифицированными программистами. Технические стандарты GDL содержат официальные стандарты Graphisoft для профессиональных разработчиков библиотек; этот документ может быть закачан из веб-сайта Graphisoft.

## НАПИСАНИЕ СКРИПТОВ

### Структура библиотечного элемента

Любой написанный на языке GDL библиотечный элемент имеет **скрипты**, которые представляют собой перечень команд GDL, с помощью которых строятся 3D-фигуры и 2D-символы. Библиотечные элементы также имеют описательную информацию для проведения количественных расчетов в ArchiCAD.

Выполнение команд **основного скрипта** предшествует выполнению любого из скриптов.

**2D-скрипт** содержит параметрическое описание 2D-чертежа. Получить доступ к **бинарным 2D-данным** библиотечного элемента (содержимое окна 2D-символа) можно с помощью команды FRAGMENT2. Если 2D-скрипт отсутствует, то бинарные 2D-данные используются для воспроизведения библиотечного элемента на плане этажа.

**3D-скрипт** содержит параметрическое описание 3D-модели. Работа с **бинарными 3D-данными** (которые генерируются при выполнении операций импорта или экспорта) возможна посредством использования команды BINARY.

**Скрипт спецификаций** содержит компоненты и описания, используемые в сметах элементов, компонент и зон. Описание **бинарных данных спецификаций** приводится в разделах *Компоненты* и *Дескрипторы* общего описания библиотечного элемента, а обращение к ним производится по команде BINARYPROP. Если скрипт спецификаций и основной скрипт отсутствуют, то в процессе построения списка смет используются бинарные данные спецификаций.

**Скрипт интерфейса пользователя** позволяет пользователям определять страницы, посредством которых можно будет вводить и редактировать значения параметров библиотечного элемента вместо использования обычного списка параметров.

В **скрипте параметров** для любого из параметров библиотечного элемента можно определить список возможных значений.

Параметр, установленный в разделе **параметров**, используется в качестве значения по умолчанию при размещении библиотечного элемента на плане этажа.

**Рисунок образца** воспроизводится в диалоговом окне установки параметров библиотечного элемента при просмотре активной библиотеки. Обращение к этому образцу производится из 2D- и 3D-скриптов по командам PICTURE и PICTURE2.

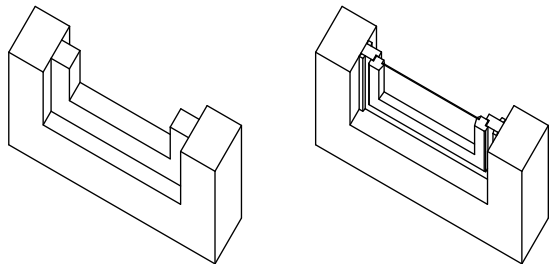
В разделе **комментариев** библиотечного элемента может храниться относящаяся к нему произвольная текстовая информация.

ArchiCAD и ArchiFM имеют свою среду написания GDL-скриптов, которая позволяет создавать скрипты, проверять их синтаксис и осуществлять визуализация описываемых объектов.

## Анализ, декомпозиция и упрощение

Каким бы сложным не являлся создаваемый Вами объект, он всегда может быть представлен в виде совокупности простых геометрических фигур. Всегда начинайте процесс создание объекта с его анализа, с выявления простейших фигур, из которых он состоит. Только после такой декомпозиции Вы можете представить объект в целом в виде команд языка GDL. Если проведенный Вами анализ был адекватным, то последующий синтез приведет к созданию именно того, что Вы хотели.

Для проведения анализа необходимы две вещи: чувство объемного восприятия и знание основ начертательной геометрии.



*Представление окна с различной степенью детализации*

Чтобы избежать непреодолимых препятствий на начальном этапе моделирования, начинайте с объектов определенных размеров, придавая им простейшую форму. По мере более глубокого понимания и четкого восприятия их структуры, создавайте более адекватные формы, пока не достигните идеального представления.

“Идеальный” не означает “сложный.” В зависимости от целей создаваемого проекта идеальный библиотечный элемент может изменяться от схематического представления и до тщательной отрисовки всех его составляющих. Окно слева на рисунке выше прекрасно подходит для его представления на генеральном плане. В свою очередь, окно справа дает более реальное и детальное представление, которое впоследствии можно воспользоваться при создании конструкторской документации проекта.

## Разработка

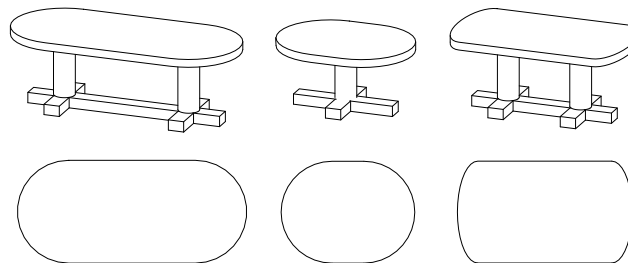
В зависимости от стоящих задач разрабатываемые Вами объекты могут существенно отличаться по своей сложности. Объекты, создаваемые для специального применения, могут быть менее сложными, чем те, которые предназначаются для универсального использования или коммерческого распространения.

Если для Вас не является принципиальным, как должен выглядеть символ на плане этажа, или если нет необходимости в параметрических изменениях такого символа, то Вы вообще можете не создавать для него параметрический 2D-скрипт.

Даже если имеется потребность в параметрических заменах

2D-символа, то все же нет абсолютной необходимости в создании параметрического 2D-скрипта. В этом случае можно изменить соответствующим образом параметры 3D-скрипта, воспроизвести объект в 3D-окне и использовать его вид сверху в качестве 2D-символа, сохранив под новым именем. Так можно создать множество подобных объектов из одного исходного.

Большинство сложных библиотечных элементов состоят из параметрических 3D-описаний и параметрических 2D-скриптов. При этом любые изменения параметров влияют не только на их трехмерные изображения, но и на их представление на плане этажа.



## Простейшие команды

Эти команды легко освоить и использовать. Они не требуют каких-либо навыков программирования, тем не менее, с их помощью можно создавать новые полезные объекты.

## Простые фигуры

Простые фигуры GDL являются теми геометрическими элементами, на основе которых строятся сложные библиотечные объекты. Они являются основными строительными кирпичиками GDL. Размещение фигуры в трехмерном пространстве производится написанием ее команды в скрипте GDL.

Команда фигуры состоит из ключевого слова, определяющего ее тип, и последовательности числовых величин или алфавитных параметров, которые задают ее размерные характеристики.

Количество таких значений зависит от фигуры.

Если это возможно, то вначале Вы можете избежать использования параметров и работать только с фиксированными значениями.

Вы можете приступить к изучению GDL со следующих команд простейших фигур:

### В 3D:

BLOCK CYLIND SPHERE PRISM

### В 2D:

LINE2 RECT2 POLY2 CIRCLE2 ARC2

## Преобразование координат

Можно себе представить, что преобразование координат - это перемещение руки в то место, где Вы хотите что-то нарисовать. С помощью преобразования координат Вы определяете расположение, ориентацию и масштаб последующей фигуры.

```
BLOCK 1, 0.5, 0.5
ADDX 1.5
ROTY 30
BLOCK 1, 0.5, 0.5
```

При желании в 3D-окне библиотечного элемента вместе с самим объектом могут быть показаны исходное (G = глобальное) и текущее (L = локальное) положения осей координат.

Простейшие преобразования координат следующие:

### В 3D:

```
ADDX ADDY ADDZ
ROTX ROTY ROTZ
```

### В 2D:

```
ADD2 ROT2
```

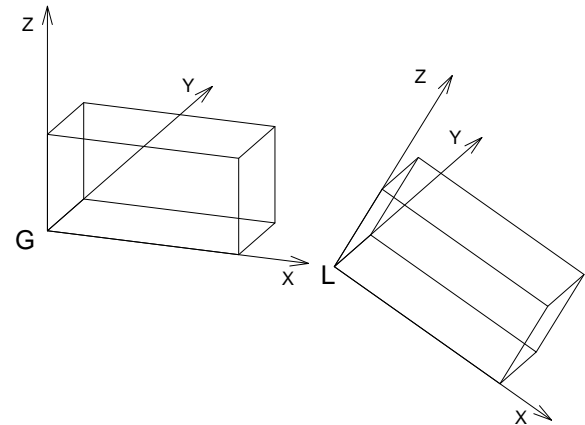
Команды ADD приводят к перемещению последующей фигуры, а команды ROT - к повороту вокруг соответствующей оси.

## Команды средней сложности

Они немного сложнее не столько потому, что требуют навыков программирования, сколько потому, что описывают более сложные фигуры или абстрактные преобразования.

### В 3D:

```
ELLIPS CONE
POLY_ LIN_ PLANE PLANE_
PRISM_ CPRISM_ SLAB SLAB_ CSLAB_
TEXT
```





**B 2D:**

```
HOTSPOT2 POLY2_ TEXT2 FRAGMENT2
```

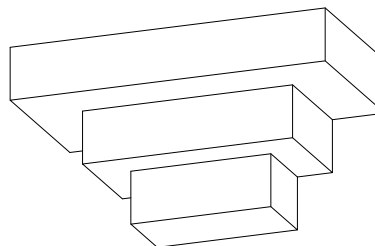
Эти команды обычно требуют определения большого количества значений по сравнению с простыми командами. Некоторые из них требуют задания значений статуса для визуализации ребер и поверхностей.

**Преобразование координат****B 3D:**

```
MULX MULY MULZ
ADD MUL ROT
```

**B 2D:**

```
MUL2
PRISM 4, 1, 3, 0,
      3, 3, -3, 3,
      -3, 0
ADDZ -1
MUL 0.666667, 0.666667, 1
PRISM 4, 1, 3, 0,
      3, 3, -3, 3,
      -3, 0
ADDZ -1
MUL 0.666667, 0.666667, 1
PRISM 4, 1, 3, 0,
      3, 3, -3, 3,
      -3, 0
```



Команды `MUL` приводят к изменению масштаба последующих фигур относительно соответствующих осей, преобразуя, например, окружности в эллипсы, а сферы в эллипсоиды. При отрицательных значениях параметров происходит зеркальное отражение последующих фигур. Команды во второй строке приводят к одновременному изменению всех трех осей координат.

## Сложные команды и дополнительные возможности

Команда данной группы являются наиболее сложными, либо вследствие сложности их геометрической формы, либо потому, что они требуют навыков программирования.

### В 3D:

BPRISM_	BWALL_	CWALL_	XWALL_
CROOF_	FPRISM_	SPRISM_	
EXTRUDE	PYRAMID	REVOLVE	RULED
SWEEP	TUBE	TUBEA	COONS
MESH	MASS		
LIGHT	PICTURE		

Здесь имеются команды, позволяющие создавать сглаженные криволинейные поверхности путем перемещения ломаной линии вдоль другой пространственной ломаной линии. Некоторые из фигур требуют задания в списке параметров значений покрытий их поверхностей.

С помощью плоскостей сечения (в форме многоугольников или фигур) Вы можете получать сложные фигуры произвольной формы, которые строятся из простых фигур. Для этого предлагаются команды CUTPLANE, CUTPOLY, CUTPOLYA, CUTSHAPE и CUTEND.

### В 2D-окне

PICTURE2	POLY2_A
SPLINE2	SPLINE2_A

## Предложения условного и безусловного перехода

```
FOR NEXT
DO WHILE ENDWHILE
REPEAT UNTIL
IF THEN ELSE ENDIF GOTO GOSUB
RETURN END EXIT
```

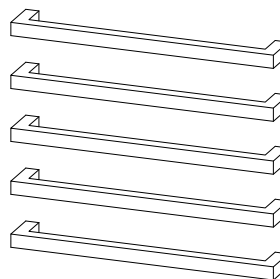
Команда данной группы хорошо известны тем, кто имеет опыт программирования, однако в их основе лежат настолько очевидные идеи, что их легко понять и непрограммистам.

Они дают возможность многократно выполнять некоторые части скриптов, позволяя тем самым при каждом выполнении описывать новые геометрические фигуры, либо принимать решения на основе ранее вычисленных значений.

```

FOR I = 1 TO 5
PRISM_ 8, 0.05,
    -0.5, 0, 15,
    -0.5, -0.15, 15,
    0.5, -0.15, 15,
    0.5, 0, 15,
    0.45, 0, 15,
    0.45, -0.1, 15,
    -.45, -0.1, 15,
    -0.45, 0, 15
ADDZ 0.2
NEXT I

```



## Параметры

Язык GDL позволяет использовать вместо фиксированных числовых значений алфавитные переменные. Это способствует созданию параметрически настраиваемых объектов. Значения таких переменных указываются в диалоговых окнах установки параметров библиотечных элементов во время работы над проектом.

## Макровывозы

При написании скрипта Вы не ограничиваетесь имеющимися командами GDL для описания фигур. Любой существующий библиотечный элемент во всей своей совокупности может стать частью создаваемой Вами GDL-фигуры. Для этого достаточно задать его имя и передать необходимые параметры точно так же, как это делается при использовании обычных команд фигур.

## Профессиональное использование GDL

После глубокого изучения описанных выше команд и получения практического опыта работы с ними Вы будете в состоянии освоить еще несколько специфических команд и возможностей языка, необходимость использования которых будет время от времени возникать у Вас.

**Примечание:** Заметим, что в связи с ограниченными возможностями памяти Вашего компьютера, могут быть ограничены длина файла, содержащего GDL-скрипт, глубина вложенности макровывозов и количество преобразований координат.

Оставшаяся часть данного пособия дает исчерпывающую информацию по приведенным выше командам GDL. Оперативную информацию о синтаксисе языка, командах и структуре их параметров можно получить в справочных файлах, имеющихся в составе Вашего программного обеспечения.

## КАК СОЗДАЮТСЯ ТРЕХМЕРНЫЕ ИЗОБРАЖЕНИЯ

3D-моделирование в ArchiCAD базируется на арифметике с плавающей запятой; это означает, что практически отсутствуют какие-либо ограничения на геометрические размеры модели. Какого бы размера не была она, степень точности представления даже ее мельчайших частей остается одной и той же.

3D-модель, которую Вы видите на экране, состоит из **геометрических примитивов**. Эти примитивы хранятся в памяти компьютера в двоичном формате и ArchiCAD генерирует их согласно плану этажа. Преобразование между этими архитектурными элементами плана этажа и двоичными 3D-данными называется **3D-преобразованием**.

Геометрическими примитивами являются следующие:

- все **вершины** создаваемых компонент;
- все **ребра**, соединяющие вершины;
- все **поверхности** многоугольников, образованных ребрами.

Совокупность примитивов, объединенных в единое целое, называется **телом**. Множество тел составляют 3D-модель. Вся трехмерная визуализация в ArchiCAD - сглаженные поверхности, отбрасывание теней, прозрачные или отражающие свет покрытия - основывается на этой структуре данных.

## Трехмерное пространство в ArchiCAD

3D-модель создается в трехмерном пространстве, которое задается осями x, y и z **основной системы координат**; начало данной системы координат называется **глобальным (основным)**.

Если Вы запускаете ArchiCAD без вызова какого-либо документа, то начало основных координат появляется в нижнем левом углу чертежного листа. Кроме того, начало основных координат определяет также нулевой уровень всех этажей проекта.

При размещении объекта в проекте его расположение на плане этажа определяется координатами x и y относительно основной системы координат. Его расположение относительно оси z может быть задано в диалоговом окне установки параметров объекта или определено непосредственно при его размещении в 3D. Это расположение выступает в качестве основы и значением по умолчанию для **локальной системы координат** объекта. Фигуры, описываемые в скрипте, будут размещаться относительно этой локальной системы координат.

## Для чего нужно преобразование координат?

Каждая фигура GDL привязывается к текущему положению локальной системы координат. Например, по команде BLOCK происходит построение прямоугольного параллелепипеда, один из углов которого располагается в начале локальных координат. Три его ребра направлены вдоль осей координат в положительную сторону. Таким образом, для команды BLOCK требуется задать три параметра, определяющие размеры этих трех ребер.

Каким же образом можно построить еще один параллелепипед, который был бы смещен и повернут относительно исходного? Этого нельзя указать с помощью параметров данной команды. Команда BLOCK таких параметров не имеет.

Решением данной проблемы является предварительное перемещение локальной системы координат в требуемое положение и последующее использование команды BLOCK. С помощью команд преобразования координат Вы можете определить смещение и/или поворот локальной системы координат. Эти преобразования не применяются к уже построенным фигурам, а оказывают влияние на построение последующих фигур.

## Интерпретатор GDL

При выполнении скрипта GDL-интерпретатор ArchiCAD определяет для библиотечного элемента его место расположения, размер, угол поворота, задаваемые пользователем параметры и возможное зеркальное отражение. Затем он перемещает локальную систему координат в требуемое положение и приступает к обработке скрипта библиотечного элемента. Всякий раз, когда в скрипте встречается команда какой-либо фигуры, интерпретатор генерирует соответствующие ей геометрические примитивы.

По завершению работы интерпретатора в памяти компьютера создается 3D-модель в двоичном формате, относительно которой, например, можно решать задачу построения реалистического изображения.

В ArchiCAD и ArchiFM помимо интерпретатора имеется еще и предкомпилятор GDL. Прежде, чем поступить на вход интерпретатора, исходный текст скрипта обрабатывается предкомпилятором, который создает удобный для компилятора код. Все это увеличивает скорость анализа скрипта. При изменении текста GDL-скрипта создается новый код.

Структуры данных, конвертируемые из файлов других форматов (например, DXF, Zoom, Alias Wavefront), запоминаются в двоичном 3D-разделе библиотечного элемента ArchiCAD. Обратиться к данному разделу можно из скрипта GDL с помощью предложения `BINARY`.

## Порядок анализа GDL-скриптов

Пользователям ArchiCAD не предоставляются никакие средства контроля за порядком анализа библиотечных элементов, размещаемых на плане этажа. Порядок анализа GDL-скриптов устанавливается на основе внутренней структуры данных; более того, команды *Отменить* и *Повторить* могут влиять на этот порядок, наряду с другими операциями изменения библиотечных элементов. Единственным исключением из правил являются специальные GDL-скрипты активной библиотеки, имена которых начинаются с **MASTER\_GDL** или **MASTEREND\_GDL**.

Скрипты, имена которых начинаются на **MASTER\_GDL**, всегда выполняются первыми в любой последовательности трехмерных преобразований, до создания разреза/фасада, до начала составления списка смет, сразу же после загрузки активной библиотеки.

Скрипты, имена которых начинаются на **MASTEREND\_GDL**, выполняются после трехмерных преобразований, после построения разреза/фасада, по завершению построения списка смет и в случае изменения активной библиотеки (после выполнения команд *Загрузить библиотеку*, *Открыть проект*, *Новый проект*, *Завершить*).

Эти скрипты не выполняются при редактировании библиотечных элементов. Если Ваша библиотека содержит один или более таких скриптов, то все они будут выполняться, однако порядок их не определен.

**MASTER\_GDL**- и **MASTEREND\_GDL**-скрипты могут включать определения реквизитов, инициализацию глобальных переменных пользователей, 3D-команды (только для 3D-модели), определения списков значений (см. *“VALUES” на стр. 185*), а также некоторые другие команды GDL. Определенные в скриптах реквизиты объединяются с текущим набором реквизитов ArchiCAD (реквизиты ArchiCAD с идентичными именами не замещаются, тогда как реквизиты, определенные в GDL, и не редактируемые в ArchiCAD, наоборот, всегда замещаются одноименными реквизитами ArchiCAD).



---

# СИНТАКСИС GDL

В этой главе описываются основные синтаксические элементы GDL, включая предложения, метки, идентификаторы, переменные и параметры. Также описываются правила типографского оформления синтаксических конструкций языка.

## Правила синтаксиса GDL

Язык GDL не чувствителен к регистру, на котором вводятся алфавитные символы. Это означает, что в нем не делаются различия между заглавными и строчными буквами. Исключения составляют строки, заключенные в кавычки. Логическим концом скрипта GDL являются либо предложения END или EXIT, либо физический конец скрипта.

## ПРЕДЛОЖЕНИЯ

Скрипт GDL состоит из предложений. Предложение может начинаться ключевым словом (определяющим фигуру GDL, преобразование координат, условный или безусловный переход), именем макроса или именем переменной, за которым следует символ "=" и арифметическое выражение.

## СТРОКИ

Предложения располагаются в строках, которые завершаются символами разделителя строк (символ конец\_строки). Запятая (,) в конце строки свидетельствует, что предложение продолжается на следующей строке. Двоеточие (:) используется для разделения предложений GDL, расположенных в одной строке. Восклицательный знак (!) свидетельствует, что за ним следует текст комментария. В любом месте скрипта можно вставлять любое количество пустых строк. Между операторами и операндами можно вставлять любое количество пробелов и знаков табуляции. После ключевого слова предложения и имени макроса обязательно должен присутствовать символ пробела или знак табуляции.

## МЕТКИ

Строка может начинаться меткой, которая используется для ссылки на последующее предложение. *Метка* - это целое число или строка символов, заключенная в кавычки, за которым следует двоеточие (:). Строковая этикетка является чувствительной к регистру. Метки должны быть уникальными. По меткам можно производить передачу управления выполнения скрипта. Такая передача производится предложениями GOTO и GOSUB.

## СИМВОЛЫ

Текст GDL-скрипта формируется из прописных и строчных букв английского алфавита, чисел и следующих символов:

<пробел> \_ (подчеркивание) ~ ! : , ; . + - \* / ^ = <> <=># ( ) [ ] { } \ @ & | (вертикальная черта) " ' ` " ' ` <конец\_строки>

## СТРОКИ СИМВОЛОВ

Любая строка символов Unicode, заключенная в кавычки (“, ’, ` , ‘), или строка символов без кавычек, которая не выступает в скрипте в качестве идентификатора с заданным значением (макрывзов, имя реквизита, имя файла). Все буквы символьной строки, не заключенной в кавычки, будут преобразованы в прописные, поэтому рекомендуется использовать кавычки. Максимально допустимая длина строки - 255 символов. В интерфейсе пользователя ArchiCAD, в отличие от GDL, используются не символы Unicode, поэтому в редакторе библиотечного элемента Вы можете использовать только те символы, которые имеются в кодовой странице Вашей системы.

Символ ‘\’ имеет специальное значение. Его смысл следующим образом зависит от следующего за ним символа:

\\	сам символ ‘\’
\n	переход на новую строку
\t	символ табуляции
\new line	продолжение символьной строки на следующей строке
\ с другими	не корректно и при этом выдается предупреждающее сообщение

*Примеры.*

```
"Это строка символов"  
"умывальник размером 1'-6"*1'-2"  
'Не используйте различные разделители'
```

## ИДЕНТИФИКАТОРЫ

*Идентификаторы* - это специальные символьные строки:

- не длиннее 255 символов;
- начинающиеся с алфавитной буквы или символов ‘\_’ или ‘~’;
- состоящие из букв, цифр и символов ‘\_’ или ‘~’;
- заглавные и строчные буквы не различаются.

Идентификаторами могут быть ключевые слова GDL, глобальные или локальные переменные или строки (имена). Имена ключевых слов и глобальных переменных определены ArchiCAD, все другие идентификаторы могут использоваться как имена переменных.

## ПЕРЕМЕННЫЕ

Программа на языке GDL оперирует числовыми переменными (определяемыми их идентификаторами), числами и строками символов.

Имеется два набора переменных: локальные и глобальные.



Все идентификаторы, не являющиеся ключевыми словами, глобальными переменными, именами реквизитов, именами макросов или именами файлов, рассматриваются как локальные переменные. Если они не инициализируются, то их значение равно 0 (целое). При макровывозах значения локальных переменных помещаются в стек. После выхода из макроса интерпретатор восстанавливает их значения.

Глобальные переменные имеют специальные зарезервированные имена (список доступных из ArchiCAD глобальных переменных приведен в *“Разное” на стр. 223*). Они не помещаются в стек при макровывозах, и их значения можно использовать в макросах. Это дает возможность запоминать в них специальные значения для их совместного использования, а также формировать коды возврата из макроса. Глобальные переменные, определенные пользователем, могут быть проинициализированы в любом скрипте. Однако они будут действовать только в последующих скриптах. Если Вы хотите, чтобы желаемый скрипт анализировался непременно первым, объявите эти переменные в библиотечном элементе MASTER\_GDL. Другие глобальные переменные могут использоваться Вами в скриптах для организации взаимодействия с ArchiCAD.

Используя команду “=”, Вы можете присвоить локальной или глобальной переменной значение числового или строкового типа.

## ПАРАМЕТРЫ

Идентификаторы, перечисленные в списке параметров библиотечного элемента, называются *параметрами*. Длина идентификатора параметра не должна превышать 32 символа. Внутри скрипта к параметрам применимы те же правила, что и к локальным переменным.

Параметры текстовых файлов GDL идентифицируются буквами A..Z.

## ПРОСТЫЕ ТИПЫ

Переменные, параметры и выражения могут быть двух простых типов: числового и строкового.

**Числовыми выражениями** являются числовые константы, числовые переменные или параметры, функции, которые возвращают числовые значения, а также любая их комбинация в операциях над числовыми выражениями. Числовые выражения могут быть целого или вещественного типа. Выражениями целого типа являются целочисленные константы, переменные или параметры, функции, которые возвращают целые значения, и любая комбинация перечисленных выше элементов, дающая в результате целое число. Выражение вещественного типа - это вещественные константы, переменные или параметры, функции, возвращающие вещественные значения, и любая комбинация перечисленных выше элементов (включая целочисленные выражения), дающая в результате вещественное число. Является ли числовое выражение целым или вещественным типом, определяется при компиляции и зависит только от используемых в нем констант, переменных, параметров и операций. Выражения целого или вещественного типа могут использоваться одинаково в тех местах, где требуется использование числового выражения, однако, если их комбинация приводит к проблеме точности представления, компилятор выводит предупреждающее сообщение (сравнение вещественных чисел или вещественных чисел с целыми числами с помощью операторов отношений '=' или '<>', или использование логических операторов AND, OR, EXOR; использование предложений IF или GOTO, когда выражение метки имеет вещественное значение).

**Строковыми выражениями** являются строковые константы, строковые переменные или параметры, функции, возвращающие в качестве результата строку символов, а также любая их комбинация в операциях, дающих в качестве результата строку символов.

## ПРОИЗВОДНЫЕ ТИПЫ

Переменные и параметры могут быть также массивами, а параметры могут быть представлены списками значений простого типа.

**Массивами** называются одномерные или двумерные таблицы числовых и/или строковых значений, каждое из которых имеет свой индекс, посредством которого и осуществляется доступ.

**Списки значений** - это наборы возможных значений числовых или строковых значений. Эти значения могут быть назначены параметрам либо в скрипте списка значений библиотечного элемента, либо в скрипте MASTER\_GDL, и появляются в списке параметров во всплывающем меню.

### [aaa]

Квадратные скобки обозначают, что заключенная в них конструкция не является обязательной (жирные квадратные скобки указывают, что необходимо произвести ввод значения так, как указано).

### {n}

Номер версии команды.

...

Предыдущая конструкция может быть повторена.

### variable

Произвольное имя переменной GDL.

### prompt

Произвольная строка символов, не содержащая кавычек.

### ЖИРНАЯ\_СТРОКА

### ЗАГЛАВНАЯ\_СТРОКА

### специальные символы

Должны вводиться так, как указано.

### другие\_последовательности\_строчных\_символов\_в\_списке\_параметров

Произвольное выражение GDL.

# ПРЕОБРАЗОВАНИЕ КООРДИНАТ

В этой главе описываются типы преобразований, имеющиеся в GDL (перемещение, масштабирование, поворот системы координат), и способы их интерпретации и управления.

## Про преобразования координат

В GDL все геометрические элементы привязаны к локальной системе координат. GDL использует правостороннюю систему координат. Например, один из углов прямоугольного параллелепипеда располагается в начале локальных координат, а три его ребра направлены вдоль осей  $x$ ,  $y$  и  $z$ .

Размещение геометрического элемента в необходимом месте производится в два приема. Во-первых, необходимо переместить локальную систему координат в требуемое место. Во-вторых, создать сам элемент. Любое перемещение, поворот и масштабирование системы координат вдоль или вокруг любой из осей называется ее **преобразованием координат**.

Преобразования запоминаются в стеке. Интерпретация скрипта начинается относительно последнего запомненного в стеке преобразования. Скрипты наследуют этот стек преобразований. Они могут вводить новые преобразования в стек, а также удалять их, причем только те, которые были созданы данным скриптом. Имеется возможность одновременно удалить одно, несколько или все преобразования, определенные в текущем скрипте. После завершения работы скрипта все его локально определенные преобразования удаляются из стека.

## ПРЕОБРАЗОВАНИЯ В ДВУМЕРНОМ ПРОСТРАНСТВЕ

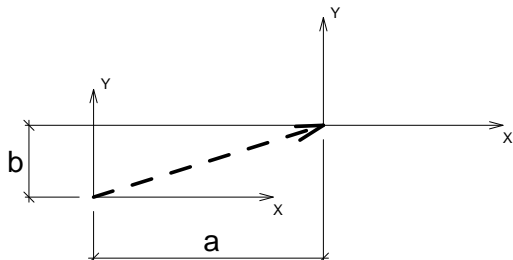
Эти команды эквивалентны командам ADD, MUL в плоскости и ROTZ в пространстве.

### ADD2

**ADD2**  $x$ ,  $y$

Пример:

ADD2  $a$ ,  $b$



## MUL2

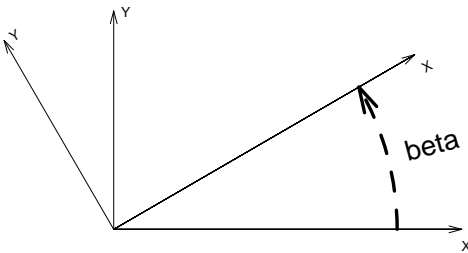
**MUL2**  $x, y$

## ROT2

**ROT2** alpha

*Пример:*

ROT2 beta



## ПРЕОБРАЗОВАНИЯ В ТРЕХМЕРНОМ ПРОСТРАНСТВЕ

### ADDX

**ADDX** dx

### ADDY

**ADDY** dy

### ADDZ

**ADDZ** dz

Перемещение локальной системы координат вдоль соответствующей оси на расстояние dx, dy или dz соответственно.

## ADD

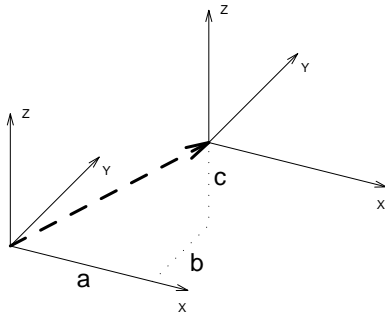
**ADD** dx, dy, dz

Равносильна командам **ADDX** dx: **ADDY** dy: **ADDZ** dz.

Приводит к созданию одного элемента в стеке преобразований, который может быть удален по команде **DEL 1**.

*Пример:*

**ADD a,b,c**



## MULX

**MULX** mx

## MULY

**MULY** my

## MULZ

**MULZ** mz

Масштабирование соответствующей оси локальных координат. Отрицательные значения mx, my, mz означают одновременное изменение направлений осей координат (зеркальное отражение).

## MUL

**MUL** mx, my, mz

Равносильна командам **MULX** mx: **MULY** my: **MULZ** mz. Приводит к созданию одного элемента в стеке преобразований, который может быть удален по команде **DEL 1**.

## ROTX

**ROTX**  $\alpha$

## ROTY

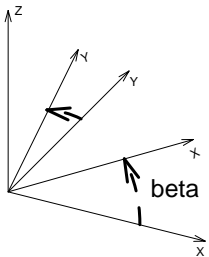
**ROTY**  $\alpha$

## ROTZ

**ROTZ**  $\alpha$

Поворот локальной системы координат вокруг соответствующей оси против часовой стрелки на  $\alpha$  градусов соответственно.

*Пример:*



ROTZ  $\beta$

## ROT

**ROT**  $x, y, z, \alpha$

Поворот локальной системы координат вокруг оси, определенной вектором  $(x,y,z)$ , против часовой стрелки на  $\alpha$  градусов. Приводит к созданию одного элемента в стеке преобразований, который может быть удален по команде DEL 1.

## XFORM

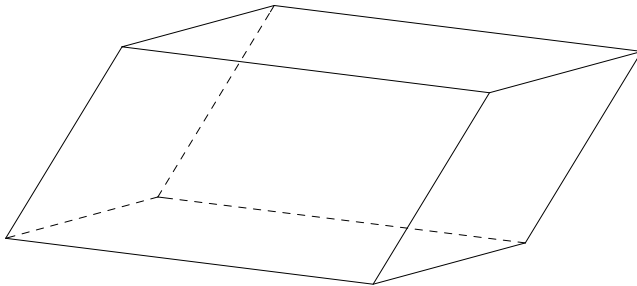
**XFORM**  $a_{11}, a_{12}, a_{13}, a_{14},$   
 $a_{21}, a_{22}, a_{23}, a_{24},$   
 $a_{31}, a_{32}, a_{33}, a_{34}$

Определяет полную матрицу преобразований. Эта команда используется в основном в автоматической генерации GDL кода. Приводит к созданию одного элемента в стеке преобразований.

$$\begin{aligned}x' &= a_{11} * x + a_{12} * y + a_{13} * z + a_{14} \\y' &= a_{21} * x + a_{22} * y + a_{23} * z + a_{24} \\z' &= a_{31} * x + a_{32} * y + a_{33} * z + a_{34}\end{aligned}$$

*Пример:*

```
A=60
B=30
XFORM 2, COS(A), COS(B)*0.6, 0,
      0, SIN(A), SIN(B)*0.6, 0,
      0, 0, 1, 0
BLOCK 1, 1, 1
```



## УПРАВЛЕНИЕ СТЕКОМ ПРЕОБРАЗОВАНИЙ

### DEL n

**DEL n** [, begin\_with]

Удаляет n последних помещенных в стек элементов.

Если параметр begin\_with не определен, удаляет n предшествующих элементов, помещенных в стек преобразований.

Устанавливается та локальная система координат, которая соответствует самому верхнему элементу в стеке.

Если параметр begin\_with определен, удаляет n следующих элементов, начиная с указанного параметром begin\_with. Нумерация начинается с 1. Если параметр begin\_with определен и его значение отрицательно, удаление элементов производится в обратном направлении.

Если число n оказывается больше количества помещенных данным скриптом преобразований, то удаляются только те из них, которые были им инициированы.

## DEL TOP

### DEL TOP

Удаляет все преобразования текущего скрипта.

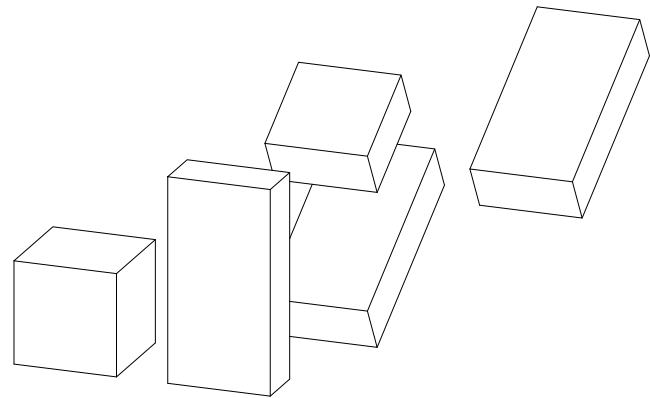
## NTR

### NTR ()

Возвращает текущее количество преобразований.

*Пример:*

```
BLOCK 1, 1, 1
ADDX 2
ADDY 2.5
ADDZ 1.5
ROTX -60
ADDX 1.5
BLOCK 1, 0.5, 2
DEL 1, 1           !Удаляет преобразование ADDX 2
BLOCK 1, 0.5, 1
DEL 1, NTR() -2   !Удаляет преобразование ADDZ 1.5
BLOCK 1, 0.5, 2
DEL -2, 3         !Удаляет преобразования ROTX -60
                  !и ADDY 2.5
BLOCK 1, 0.5, 2
```





# ПРОСТРАНСТВЕННЫЕ ФИГУРЫ

В этой главе рассматриваются все команды GDL создания пространственных фигур, начиная от самых простых и до группы команд построения сложных фигур из ломаных линий. Здесь также представлены элементы визуализации (источники света, рисунки), а также определение пространственного текста. Более того, детально обсуждаются примитивные элементы внутренней структуры 3D-данных, состоящие из вершин, векторов, ребер и тел, а также интерпретация двоичных данных и общие правила использования плоскостей сечения.

## ОСНОВНЫЕ ПРОСТРАНСТВЕННЫЕ ФИГУРЫ

### BLOCK

**BLOCK**  $a, b, c$

### BRICK

**BRICK**  $a, b, c$

Прямоугольный параллелепипед, первый угол которого расположен в начале локальных координат, а ребра имеют размеры  $a$ ,  $b$  и  $c$  и направлены соответственно вдоль осей  $x$ ,  $y$  и  $z$ . Нулевые значения параметров приводят к созданию вырожденного параллелепипеда (прямоугольника или линии).

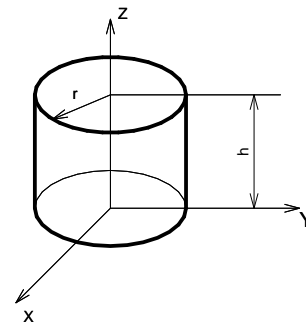
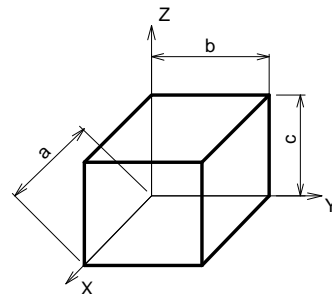
Ограничения на параметры:

$a, b, c \geq 0$

### CYLIND

**CYLIND**  $h, r$

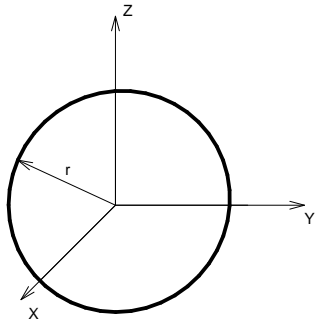
Прямой цилиндр вдоль оси  $z$ , с высотой  $h$  и радиусом  $r$ , проведенным из начала координат. Если  $h=0$ , то создается окружность в плоскости  $x-y$ . Если  $r=0$ , то создается линия вдоль оси  $z$ .



## SPHERE

**SPHERE**  $r$

Сфера с центром в начале координат и радиусом  $r$ .



## ELLIPS

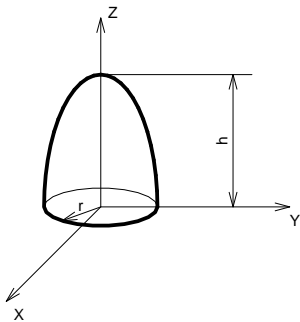
**ELLIPS**  $h, r$

Полуэллипсоид. Основание находится в плоскости  $x$ - $y$  и представляет собой окружность с центром в начале координат и радиусом  $r$ . Длина полуоси, расположенной вдоль оси  $z$ , равна  $h$ .

*Пример:*

ELLIPS  $r, r$

! полусфера



## CONE

**CONE**  $h, r1, r2, \alpha1, \alpha2$

Усеченный конус. Углы  $\alpha1$  и  $\alpha2$  указывают наклон нижней и верхней секущих плоскостей относительно оси  $z$ ;  $r1$  и  $r2$  являются радиусами нижней и верхней окружностей, а  $h$  - высота вдоль оси  $z$ .

Если  $h=0$ , то углы  $\alpha1$  и  $\alpha2$  не используются, и в плоскости  $x$ - $y$  создается круговое кольцо.

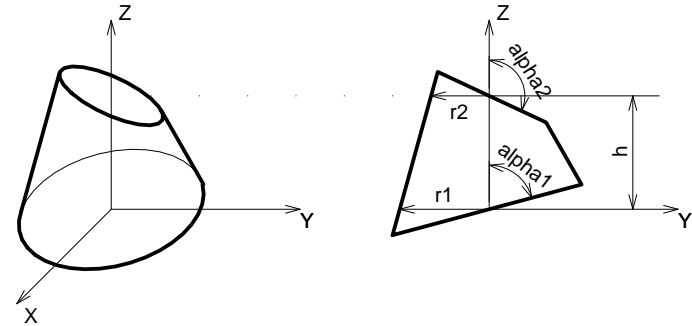
$\alpha1$  и  $\alpha2$  измеряются в градусах.

*Ограничения на параметры:*

$0 < \alpha1 < 180^\circ$  и  $0 < \alpha2 < 180^\circ$

*Пример:*

CONE  $h, r, 0, 90, 90$  ! прямой конус



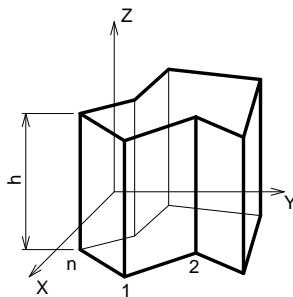
## PRISM

**PRISM**  $n, h, x1, y1, \dots, xn, yn$

Прямая призма с многоугольным основанием в плоскости  $x$ - $y$  (см. параметры предложений *"POLY"* на стр. 62 и *"POLY\_"* на стр. 62). Высота вдоль оси  $z$  равна  $\text{abs}(h)$ . Может также использоваться отрицательное значение  $h$ . В данном случае второе основание призмы расположено ниже плоскости  $x$ - $y$ .

*Ограничения на параметры:*

$n \geq 3$



## PRISM\_

**PRISM\_** *n*, *h*, *x1*, *y1*, *s1*, ... *xn*, *yn*, *sn*

Аналогично CPRISM\_ за исключением того, что любые из горизонтальных ребер и боковых граней могут отсутствовать.

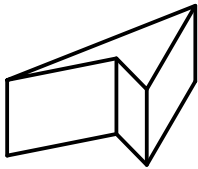
*Ограничения на параметры:*

*n* >= 3

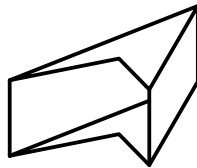
*si*: код статуса, который позволяет контролировать видимость ребер многоугольника и боковых граней. Вы также можете определить отверстия и создавать сегменты и дуги в ломаной линии с помощью специальных ограничений.

См. "*Коды статусов*" на стр. 141 для детального ознакомления.

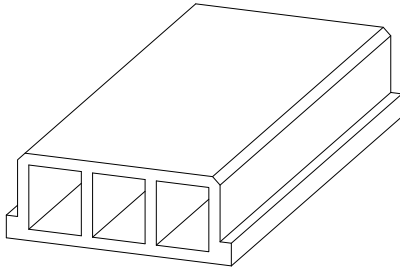
*Примеры:*



```
PRISM_ 4,1,
0,0,15,
1,1,15,
2,0,15,
1,3,15
```



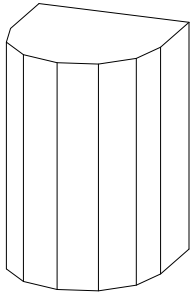
```
PRISM_ 4,1,
0,0,7,
1,1,5,
2,0,15,
1,3,15
```



```

ROTX 90
PRISM_ 26, 1.2,
    0.3, 0, 15,
    0.3, 0.06, 15,
    0.27, 0.06, 15,
    0.27, 0.21, 15,
    0.25, 0.23, 15,
    -0.25, 0.23, 15,
    -0.27, 0.21, 15,
    -0.27, 0.06, 15,
    -0.3, 0.06, 15,
    -0.3, 0, 15,
    0.3, 0, -1,           !конец контура
    0.10, 0.03, 15,
    0.24, 0.03, 15,
    0.24, 0.2, 15,
    0.10, 0.2, 15,
    0.10, 0.03, -1,      !конец первого отверстия
    0.07, 0.03, 15,
    0.07, 0.2, 15,
    -0.07, 0.2, 15,
    -0.07, 0.03, 15,
    0.07, 0.03, -1,     !конец второго отверстия
    -0.24, 0.03, 15,
    -0.24, 0.2, 15,
    -0.1, 0.2, 15,
    -0.1, 0.03, 15,
    -0.24, 0.03, -1     !конец третьего отверстия

```



j7 = 0

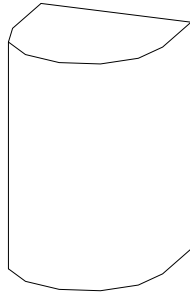
R=1

H=3

```
PRISM      9,          H,
           -R,         R,          15,
           COS (180) *R, SIN (180) *R, 15,
           COS (210) *R, SIN (210) *R, 15,
           COS (240) *R, SIN (240) *R, 15,
           COS (270) *R, SIN (270) *R, 15,
           COS (300) *R, SIN (300) *R, 15,
           COS (330) *R, SIN (330) *R, 15,
           COS (360) *R, SIN (360) *R, 15,
           R,          R,          15
```

ADDX 5

```
PRISM      9,          H,
           -R,         R,          15,
           COS (180) *R, SIN (180) *R, 64+15,
           COS (210) *R, SIN (210) *R, 64+15,
           COS (240) *R, SIN (240) *R, 64+15,
           COS (270) *R, SIN (270) *R, 64+15,
           COS (300) *R, SIN (300) *R, 64+15,
           COS (330) *R, SIN (330) *R, 64+15,
           COS (360) *R, SIN (360) *R, 64+15,
           R,          R,          15
```



j7 = 1

## CPRISM\_

**CPRISM\_** top\_material, bottom\_material, side\_material,  
n, h, x1, y1, s1, ... xn, yn, sn

Является обобщением предложения PRISM\_. Первые три параметра используются для указания наименования/индекса покрытия верхней, нижней и боковой поверхностей. Все остальные параметры аналогичны параметрам предложения PRISM\_.

*Ограничения на параметры:*

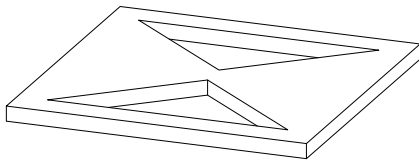
n >= 3

*См. также “Определение покрытия” на стр. 163.*

si: код статуса, который позволяет контролировать видимость ребер многоугольника и боковых граней. Также можно определить отверстия и создавать сегменты и дуги в ломаной линии с помощью специальных ограничений.

*См. “Коды статусов” на стр. 141 для детального ознакомления.*

*Пример:*



```
CPRISM_ "Железо", 0, T_,      !"Железо" - это ранее определенное покрытие
                                !0 - универсальное покрытие.
                                !T_ - глобальная переменная
                                !(индекс покрытия)

13, 0.2,
0, 0, 15,
2, 0, 15,
2, 2, 15,
0, 2, 15,
0, 0, -1,                      ! конец контура
0.2, 0.2, 15,
1.8, 0.2, 15,
1.0, 0.9, 15,
0.2, 0.2, -1,                  ! конец первого отверстия
0.2, 1.8, 15,
1.8, 1.8, 15,
1.0, 1.1, 15,
0.2, 1.8, -1                  ! конец второго отверстия
```

## BPRISM\_

**BPRISM\_** top\_material, bottom\_material, side\_material,  
n, h, radius, x1, y1, s1, ... xn, yn, sn

Сглаженная криволинейная призма, имеющая ту же структуру данных, что и CPRISM\_. Единственным дополнительным параметром является radius.

Получается из соответствующей CPRISM\_ путем изгибания плоскости x-y вокруг цилиндра, касающегося данной плоскости. Ребра вдоль оси x преобразуются в дуги. Ребра вдоль оси y остаются горизонтальными. Ребра вдоль оси z принимают радиальное направление.

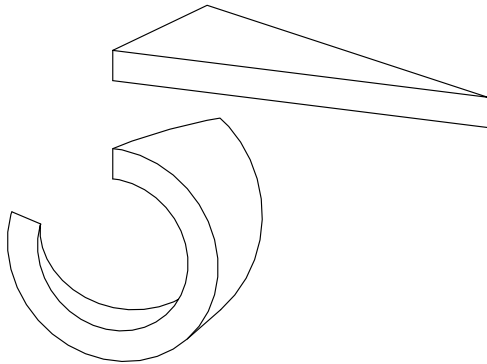
См. *"BWALL\_" на стр. 49* для детального ознакомления.

si: код статуса, который позволяет контролировать видимость ребер многоугольника и боковых граней. Вы также можете определить отверстия и создавать сегменты и дуги в ломаной линии с помощью специальных ограничений.

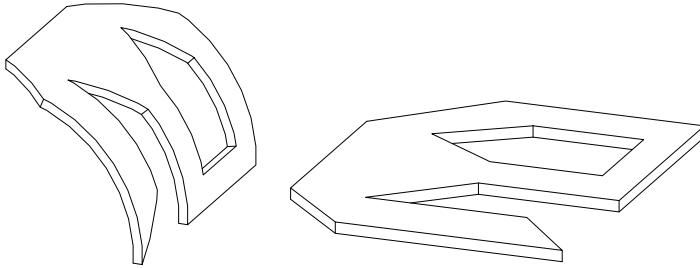
См. *"Коды статусов" на стр. 141* для детального ознакомления.

Пример (BPRISM\_ и соответствующей CPRISM\_):

```
BPRISM_ "Стекло", "Стекло", "Стекло",
        3, 0.4, 1,           ! радиус = 1
        0, 0, 15,
        5, 0, 15,
        1.3, 2, 15
```







```

BPRISM_ "Бетон", "Бетон", "Бетон",
  17, 0.3, 5,
  0, 7.35, 15,
  0, 2, 15,
  1.95, 0, 15,
  8, 0, 15,
  6.3, 2, 15,
  2, 2, 15,
  4.25, 4, 15,
  8, 4, 15,
  8, 10, 15,
  2.7, 10, 15,
  0, 7.35, -1,
  4, 8.5, 15,
  1.85, 7.05, 15,
  3.95, 5.6, 15,
  6.95, 5.6, 15,
  6.95, 8.5, 15,
  4, 8.5, -1

```

## **FPRISM\_**

```

FPRISM_ top_material, bottom_material, side_material, hill_material,
  n, thickness, angle, hill_height,
  x1, y1, s1,
  ...
  xn, yn, sn

```

Аналогично PRISM\_ но имеет три дополнительных параметра: hill\_material, angle и hill\_height parameters. Усеченная часть призмы присоединяется к верху правильной призмы.

hill\_material: покрытие боковой поверхности усеченной части призмы.

angle: угол наклона боковых граней усеченной части призмы. Ограничение:  $0 \leq \text{angle} < 90$ . Если  $\text{angle} = 0$ , в ортогональной проекции ребра боковой поверхности усеченной части образуют четверть круга с текущим разрешением (см. команды *"RADIUS"* на стр. 154, *"RESOL"* на стр. 155 и *"TOLER"* на стр. 156).

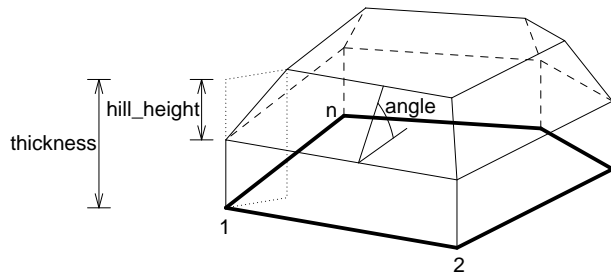
hill\_height: высота усеченной части. Параметр thickness представляет высоту всей фигуры FPRISM\_ в целом.

Ограничения на параметры:

$n \geq 3$ ,  $\text{hill\_height} < \text{thickness}$

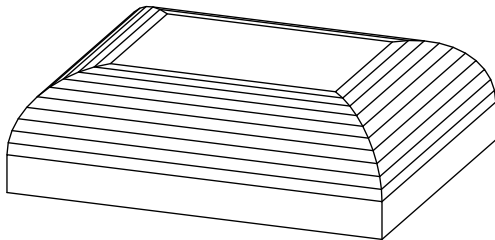
si: код статуса, который позволяет контролировать видимость ребер многоугольника и боковых граней. Можно также определить отверстия и создавать сегменты и дуги в ломаной линии с помощью специальных ограничений.

См. *"Коды статусов"* на стр. 141 для детального ознакомления.



Примеры.

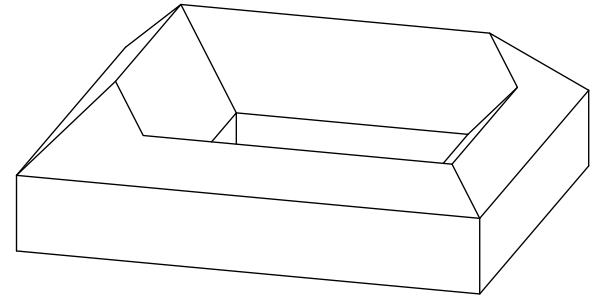
```
RESOL 10
FPRISM_ "Черепица кровельная", "Кирпич красный", "Кирпич облицовочный", "Черепица кровельная",
4, 1.5, 0, 1.0, !угол = 0
0, 0, 15,
5, 0, 15,
5, 4, 15,
0, 4, 15
```



```

FPRISM "Черепица кровельная", "Кирпич красный", "Кирпич
облицовочный", "Черепица кровельная",
    10, 2, 45, 1,
    0, 0, 15,
    6, 0, 15,
    6, 5, 15,
    0, 5, 15,
    0, 0, -1,
    1, 2, 15,
    4, 2, 15,
    4, 4, 15,
    1, 4, 15,
    1, 2, -1

```



## HPRISM\_

```

HPRISM_ top_mat, bottom_mat, side_mat,
    hill_mat,
    n, thickness, angle, hill_height, status,
    x1, y1, s1,
    ...,
    xn, yn, sn

```

Аналогично FPRISM\_, но с дополнительным параметром, контролирующим видимость ребер усеченной части.

status: управляет видимостью ребер усеченной части:

0: все ребра усеченной части видимы (FPRISM\_).

1: ребра усеченной части невидимы.

## SPRISM\_

```

SPRISM_ top_material, bottom_material, side_material,
    n, xb, yb, xe, ye, h, angle,
    x1, y1, s1, ... xn, yn, sn

```

Является расширением предложения CPRISM\_. Верхняя грань может быть не параллельна плоскости x-y. Определение верхней грани аналогично определению наклонного ската крыши в предложении CROOF\_. Высота призмы определяется относительно базовой линии (линии привязки). Верхняя и нижняя грани призмы никогда не пересекаются.

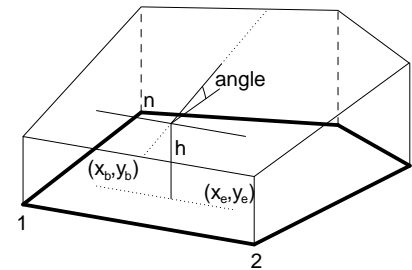
*Дополнительные параметры*

$x_b, y_b, x_e, y_e$ : начальная и конечная точки базовой линии (вектора),

$angle$ : угол поворота верхней грани вокруг базовой линии, в градусах (против часовой стрелки),

$si$ : код статуса, который позволяет контролировать видимость ребер многоугольника и боковых граней. Вы также можете определить отверстия и создавать сегменты и дуги в ломаной линии с помощью специальных ограничений.

См. “Коды статусов” на стр. 141 для детального ознакомления.



**Примечание:** Все вычисляемые значения z-координат вершин верхней грани призмы должны быть положительными или 0.

*Пример:*

```
SPRISM_ 'Трава', 'Земля', 'Земля',
6,
0, 0, 11, 6, 2, -10.0,
0, 0, 15,
10, 1, 15,
11, 6, 15,
5, 7, 15,
4.5, 5.5, 15,
1, 6, 15
```

```
SPRISM_{2} top_material, bottom_material, side_material, n,
xtb, ytb, xte, yte, topz, tangle,
xbb, ybb, xbe, ybe, bottomz, bangle,
x1, y1, s1, mat1,
...
xn, yn, sn, matn
```

Расширение предложения SPRISM\_ с дополнительной возможностью, чтобы верхняя и нижняя грани были непараллельными плоскости x-y. Определение граней аналогично определению граней в предложении CROOF\_. Верх и низ призмы определяется согласно базовой линии (линии привязки). Верхняя и нижняя грани призмы никогда не пересекаются.

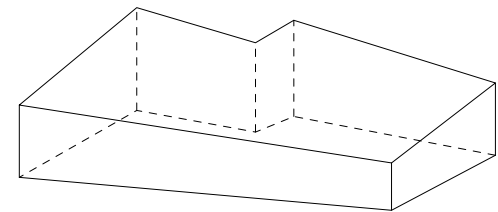
*Дополнительные параметры:*

$x_{tb}, y_{tb}, x_{te}, y_{te}$ : координаты начальной и конечной точек базовой линии (вектора) верхней грани,

$topz$ : уровень 'z' базовой линии верхней грани,

$tangle$ : угол поворота верхней грани вокруг базовой линии, в градусах (против часовой стрелки),

$x_{bb}, y_{bb}, x_{be}, y_{be}$ : координаты начальной и конечной точек базовой линии (вектора) нижней грани,



bottomz: уровень 'z' базовой линии нижней грани,

bangle: угол поворота нижней грани вокруг базовой линии, в градусах (против часовой стрелки),

si: код статуса, который позволяет контролировать видимость ребер многоугольника и боковых граней. Вы также можете определить отверстия и создавать сегменты и дуги в ломаной линии с помощью специальных ограничений.

См. *“Коды статусов” на стр. 141* для детального ознакомления.

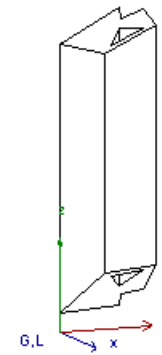
mat: ссылка на покрытие, которая позволяет покрытием боковых поверхностей.

*Пример:*

```
SPRISM_{2} 'Трава', 'Земля', 'Земля', 11,
  0,  0,  11,  0,  30, -30.0,
  0,  0,   0, 11,   2,  30.0,

  0,    0,  15, IND (MATERIAL, 'C10'),
 10,   1,  15, IND (MATERIAL, 'C11'),
 11,   6,  15, IND (MATERIAL, 'C12'),
  5,   7,  15, IND (MATERIAL, 'C13'),
  4,   5,  15, IND (MATERIAL, 'C14'),
  1,   6,  15, IND (MATERIAL, 'C10'),

  0,    0,  -1, IND (MATERIAL, 'C15'),
  9,   2,  15, IND (MATERIAL, 'C15'),
 10,   5,  15, IND (MATERIAL, 'C15'),
  6,   4,  15, IND (MATERIAL, 'C15'),
  9,   2,  -1, IND (MATERIAL, 'C15')
```



## SLAB

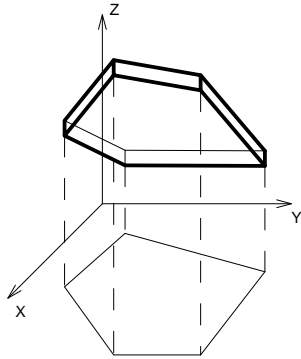
**SLAB**  $n$ ,  $h$ ,  $x_1$ ,  $y_1$ ,  $z_1$ , ...  $x_n$ ,  $y_n$ ,  $z_n$

Косая призма. Боковые грани перпендикулярны плоскости  $x$ - $y$ . Основаниями являются плоские многоугольники, повернутые относительно оси, параллельной плоскости  $x$ - $y$ . Может также использоваться отрицательное значение  $h$ . В этом случае второе основание располагается под первым.

Проверка того, что многоугольники основания являются действительно плоскими, не производится. Вершины многоугольников, которые не лежат в соответствующих плоскостях, приводят к некорректному отбрасыванию теней и построению реалистических изображений.

*Ограничения на параметры:*

$n \geq 3$



## SLAB\_

**SLAB\_**  $n$ ,  $h$ ,  $x_1$ ,  $y_1$ ,  $z_1$ ,  $s_1$ , ...  $x_n$ ,  $y_n$ ,  $z_n$ ,  $s_n$

Аналогично SLAB за исключением того, что любые ребра и поверхности боковых многоугольников могут быть опущены. В этом смысле это предложение аналогично предложению PRISM\_.

$s_i$ : код статуса, который позволяет контролировать видимость ребер многоугольника и боковых граней. Вы также можете определить отверстия и создавать сегменты и дуги в ломаной линии с помощью специальных ограничений.

См. *“Коды статусов”* на стр. 141 для детального ознакомления.

## CSLAB\_

**CSLAB\_** top\_material, bottom\_material, side\_material,  
n, h, x1, y1, z1, s1, ... xn, yn, zn, sn

Расширение предложения SLAB\_; первые три параметра используются для указания имени/индекса покрытия верхней, боковых и нижней поверхностей. Другие параметры такие же, как и в предложении SLAB\_.

См. также *“Определение покрытия”* на стр. 163, *“Запросы”* на стр. 245 > и описание функции IND в *“Разное”* на стр. 223.

si: код статуса, который позволяет контролировать видимость ребер многоугольника и боковых граней. Вы также можете определить отверстия и создавать сегменты и дуги в ломаной линии с помощью специальных ограничений.

См. *“Коды статусов”* на стр. 141 для детального ознакомления.

## CWALL\_

**CWALL\_** left\_material, right\_material, side\_material,  
height, x1, x2, x3, x4, t,  
mask1, mask2, mask3, mask4,  
n,  
x\_start1, y\_low1, x\_end1, y\_high1, frame\_shown1,  
...  
x\_startn, y\_lown, x\_endn, y\_highn, frame\_shownn,  
m,  
a1, b1, c1, d1,  
...  
am, bm, cm, dm

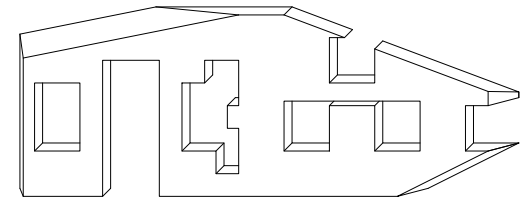
left\_material, right\_material, side\_material:

наименование/индекс покрытия для левой, правой и торцевых поверхностей. (Левая и правая поверхности определяются согласно направлению оси x.)

См. также *“Определение покрытия”* на стр. 163 и *“Запросы”* на стр. 245 > описание функции IND.

Линия привязки стены всегда преобразуется таким образом, чтобы совпадать с осью x. Боковые поверхности стены параллельны плоскости x-z.

height: высота стены относительно ее основания.



$x_1, x_2, x_3, x_4$ : проекции на ось  $x$  конечных точек стены, расположенных в плоскости  $x$ - $y$  (см. рис. ниже). Если стена стоит сама по себе, то  $x_1 = x_4 = 0, x_2 = x_3 =$  длина стены,

$t$ : толщина стены.

Если стена расположена справа от оси  $x$ , то  $t < 0$ .

Если стена расположена слева от оси  $x$  (отверстия), то  $t > 0$ ,

Если стена представляется в виде многоугольника с 'рамками' на месте отверстий, то  $t = 0$ .

$mask_1, mask_2, mask_3, mask_4$ : управляют присутствием ребер и торцевых поверхностей.

$$mask_i = j_1 + 2*j_2 + 4*j_3 + 8*j_4$$

где  $j_1, j_2, j_3, j_4$  могут принимать значения 0 или 1.

Значения  $j_1, j_2, j_3, j_4$  указывают, присутствуют (1) или отсутствуют (0) соответствующие ребра и поверхности.

$n$ : количество проемов в стене.

$x\_start_i, y\_low_i, x\_end_i, y\_high_i$ : координаты проемов согласно рис. ниже.

$frame\_shown_i$  values:

1 - ребра проема присутствуют;

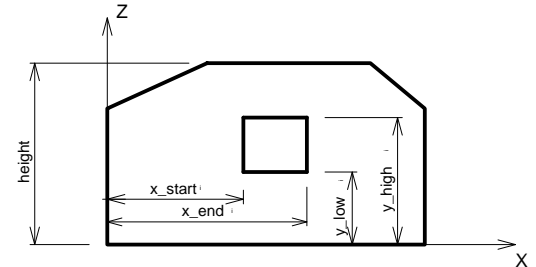
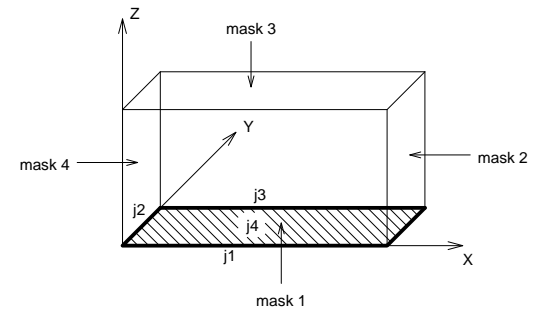
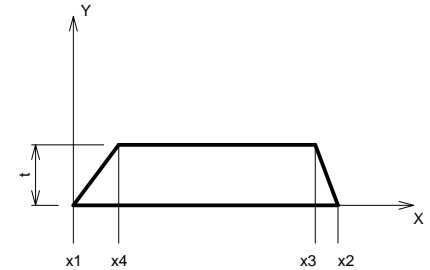
0 - ребра проема отсутствуют.

Отрицательные значения управляют присутствием каждого отдельного ребра проема.

$$frame\_shown_i = -(1*j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8),$$

где  $j_1, j_2, \dots, j_8$  могут быть 0 или 1. Числа от  $j_1$  и до  $j_4$  управляют присутствием или отсутствием ребер проема на левой стороне поверхности стены, а числа от  $j_5$  и до  $j_8$  - на правой, как это изображено на рисунке ниже.

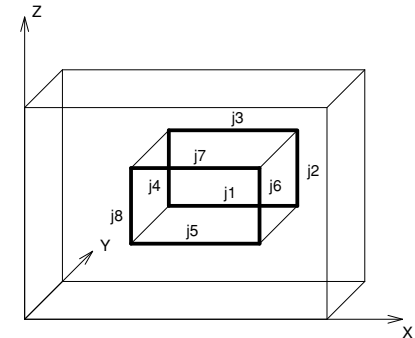
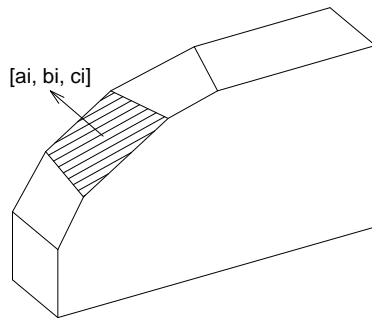
Ребро проема, перпендикулярное поверхности стены, присутствует, если присутствуют ребра, выходящие из его обоих концов.





$m$ : количество плоскостей сечения.

$a_i, b_i, c_i, d_i$ : коэффициенты следующего уравнения плоскости сечения:  $[a_i*x + b_i*y + c_i*z = d_i]$ . Те части стены, которые расположены с положительной стороны сечения (то есть  $a_i*x + b_i*y + c_i*z > d_i$ ) будут отброшены.



## BWALL\_

```

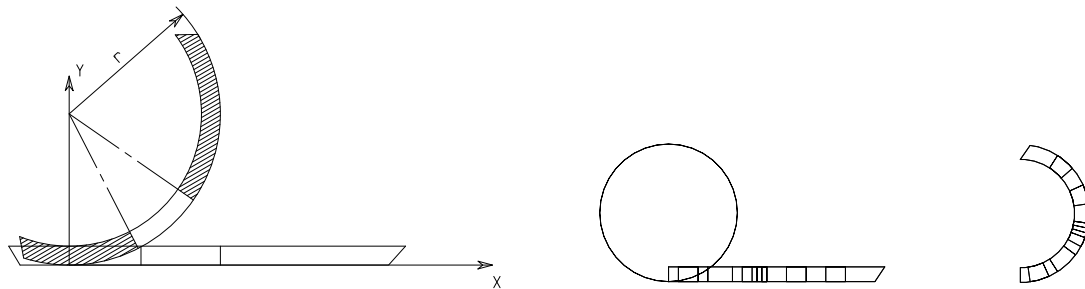
BWALL_ left_material, right_material, side_material,
height, x1, x2, x3, x4, t, radius,
mask1, mask2, mask3, mask4,
n,
x_start1, y_low1, x_end1, y_high1, frame_shown1,
...
x_startn, y_lown, x_endn, y_highn, frame_shownn,
m,
a1, b1, c1, d1,
...
am, bm, cm, dm

```

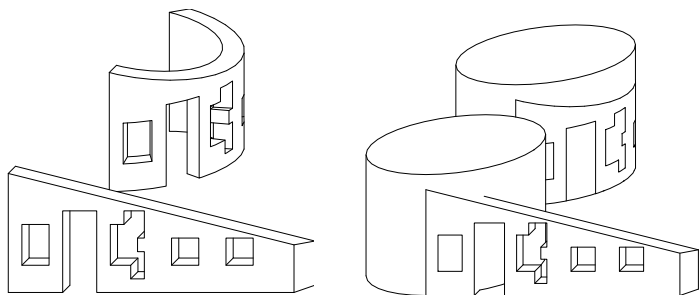
Сглаженная криволинейная стена, имеющая ту же структуру данных, что и прямолинейная стена `CWALL_`. Единственным дополнительным параметром является `radius`.

Получается из соответствующей `CWALL_` путем изгибания плоскости  $x$ - $z$  вокруг цилиндра, касающегося данной плоскости. Ребра вдоль оси  $x$  преобразуются в дуги, ребра вдоль оси  $y$  принимают радиальное направление. Ребра вдоль оси  $z$  остаются вертикальными. Кривизна аппроксимируется согласно текущему разрешению (см. команды: *"RADIUS"* на стр. 154, *"RESOL"* на стр. 155 и *"TOLER"* на стр. 156).

См. также *"CWALL\_"* на стр. 47 для детального ознакомления.



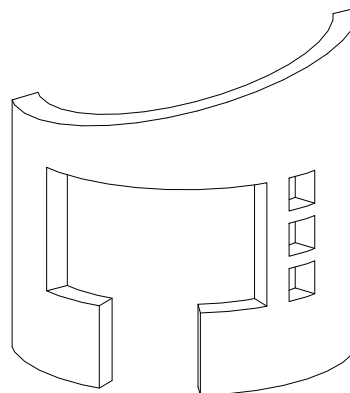
Пример: BWALL\_ и соответствующей CWALL\_.



ROTZ -60

```

BWALL_ 1, 1, 1,
        4, 0, 6, 6, 0,
        0.3, 2,
        15, 15, 15, 15,
        5,
        1, 1, 3.8, 2.5, -255,
        1.8, 0, 3, 2.5, -255,
        4.1, 1, 4.5, 1.4, -255,
        4.1, 1.55, 4.5, 1.95, -255,
        4.1, 2.1, 4.5, 2.5, -255,
        1, 0, -0.25, 1, 3
    
```



## XWALL\_

```

XWALL_ left_material, right_material, vertical_material, horizontal_material,
height, x1, x2, x3, x4,
y1, y2, y3, y4,
t, radius,
log_height, log_offset,
mask1, mask2, mask3, mask4,
n,
x_start1, y_low1, x_end1, y_high1,
frame_shown1,
...
x_startn, y_lown, x_endn, y_highn,
frame_shownn,
m,
a1, b1, c1, d1,
...
am, bm, cm, dm,
status

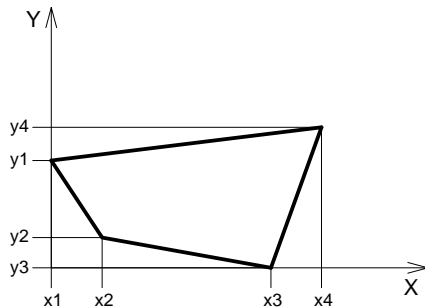
```

Обобщение определение стены на основе такой же структуры данных, как и в элементе BWALL\_.

*Дополнительные параметры*

vertical\_material, horizontal\_material: имя или индекс покрытий вертикальной/ горизонтальной поверхностей,

y1, y2, y3, y4: координаты проекции вершин стены на плоскость x-y (см рисунок ниже).



log\_height, log\_offset: дополнительные параметры, позволяющие строить стены из бревен. Действуют только для прямолинейных стен.

status: управляет свойствами бревенчатой стены.

status = j1 + 2\*j2 + 4\*j3 + 32\*j6 + 64\*j7 + 128\*j8 + 256\*j9

j1: применить покрытие правой стороны к горизонтальным ребрам.

j2: применить покрытие левой стороны к горизонтальным ребрам.

j3: начать с половины бревна,

j6: подогнать текстуру к ребрам стены,

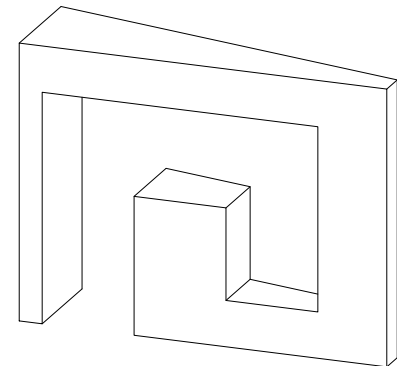
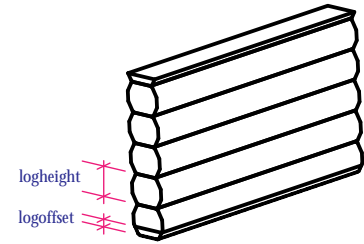
j7: двойной радиус на стороне скругления.

j8: квадратное бревно на правой стороне.

j9: квадратное бревно на левой стороне.

*Пример:*

```
XWALL_ "Побелка", "Побелка", "Побелка", "Побелка",
3.0,
0.0, 4.0, 4.0, 0.0,
0.0, 0.0, 0.3, 1.2,
1.2, 0.0,
0.0, 0.0,
15, 15, 15, 15,
3,
0.25, 0.0, 1.25, 2.5, -255,
1.25, 1.5, 2.25, 2.5, -255,
2.25, 0.5, 3.25, 2.5, -255, 0
```



**XWALL\_{2}**

```

XWALL_{2} left_material, right_material, vertical_material, horizontal_material,
height, x1, x2, x3, x4,
y1, y2, y3, y4,
t, radius,
log_height, log_offset,
mask1, mask2, mask3, mask4,
n,
x_start1, y_low1, x_end1, y_high1,
sill_depth1, frame_shown1,
...
x_startn, y_lown, x_endn, y_highn,
sill_depthn, frame_shownn,
m,
a1, b1, c1, d1,
...
am, bm, cm, dm,
status

```

Расширенное определение стены на основе той же структуры данных, что у элемента XWALL\_.

*Дополнительные параметры*

silldepthi: логическая глубина подоконника проема. Если бит j9 параметра frame\_showni установлен, то боковые покрытия стены обволакивают многоугольники отверстия; при этом параметр silldepthi определяет линию разделения между ними.

$$\text{frame\_showni} = -(1*j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8 + 256*j9 + 512*j10)$$

Имеется два дополнительных значения, управляющих обволакивание покрытием. Смысл значений j1, j2 ... j8 такой же, как и в командах CWALL\_ и XWALL\_. Значение j9 управляет покрытием многоугольников проема. Если j9 равно 1, то проем наследует боковые покрытия стены. Значение j10 управляет формой линии-разделителя между покрытиями проема в верхнем и нижнем

многоугольнике проема, когда стена является криволинейной. Если значение  $j10$  равно 1, то линия-разделитель будет прямой: в противном случае - криволинейной.

*Пример:*

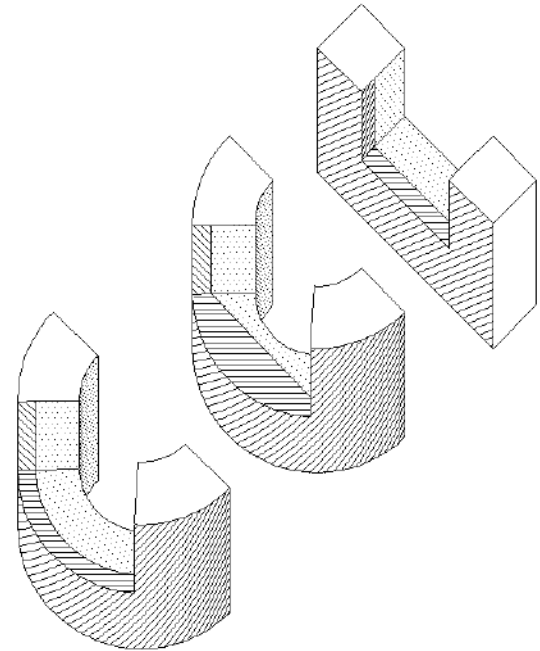
```

ROTZ 90
xWALL_{2} "C13", "C11", "C12", "C12",
  2, 0, 4, 4, 0,
  0, 0, 1, 1,
  1, 0,
  0, 0,
  15, 15, 15, 15,
  1,
  1, 0.9, 3, 2.1, 0.3, -(255 + 256),
  0,
  0

BODY -1
DEL 1
ADDX 2
xWALL_{2} "C13", "C11", "C12", "C12",
  2, 0, 2 * PI, 2 * PI, 0,
  0, 0, 1, 1,
  0, 0,
  15, 15, 15, 15,
  1,
  1.6, 0.9, 4.6, 2.1, 0.3, -(255 + 256),
  0,
  0

BODY -1

ADDX 4
xWALL_{2} "C13", "C11", "C12", "C12",
  2, 0, 2 * PI, 2 * PI, 0,
  0, 0, 1, 1,
  1, 2,
  0, 0,
  15, 15, 15, 15,
  1,
  1.6, 0.9, 4.6, 2.1, 0.3, -(255 + 256 + 512),
  0,
  0
    
```



## BEAM

**BEAM** left\_material, right\_material, vertical\_material, top\_material, bottom\_material, height, x1, x2, x3, x4, y1, y2, y3, y4, t, mask1, mask2, mask3, mask4

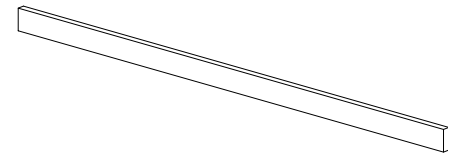
Определение балки. Параметры являются аналогичными параметрам элемента XWALL\_.

*Дополнительные параметры*

top\_material, bottom\_material: покрытия сверху и снизу.

*Пример:*

```
BEAM 1, 1, 1, 1, 1,
      0.3, 0.0, 7.0, 7.0, 0.0,
      0.0, 0.0, 0.1, 0.1, 0.5,
      15, 15, 15, 15
```



## CROOF\_

**CROOF\_** top\_material, bottom\_material, side\_material, n, xb, yb, xe, ye, height, angle, thickness, x1, y1, alpha1, s1, ..., xn, yn, alphan, sn

Наклонный скат крыши с задаваемыми углами боковых поверхностей.

top\_material, bottom\_material, side\_material: наименование/индекс покрытия для верхней, нижней и боковых поверхностей.

n: количество вершин в многоугольнике крыши.

xb, yb, xe, ye: базовая линия (вектор).

height: высота крыши в точке базовой линии (нижняя поверхность).

angle: угол поворота плоскости крыши вокруг базовой линии, имеющей заданную ориентацию. Измеряется в градусах против часовой стрелки.

thickness: толщина крыши, измеряемая перпендикулярно плоскости крыши.

$x_i, y_i$ : the координаты вершин нижнего многоугольника крыши.

$\alpha_i$ : угол между боковой поверхностью крыши, проходящей вдоль  $i$ -го ребра, и плоскостью, перпендикулярной плоскости крыши.

Ограничение:  $-90^\circ < \alpha_i < 90^\circ$ . Если смотреть по направлению ребра правильно ориентированного многоугольника крыши, то угол поворота против часовой стрелки является положительным.

Ребра многоугольника крыши имеют правильную ориентацию, если, глядя сверху, контур крыши обходится против часовой стрелки, а отверстия в крыше - по часовой стрелке.

$s_i$ : код статуса, который позволяет контролировать видимость ребер многоугольника и боковых граней. Вы также можете определить отверстия и создавать сегменты и дуги в ломаной линии с помощью специальных ограничений.

См. "Коды статусов" на стр. 141 для детального ознакомления.

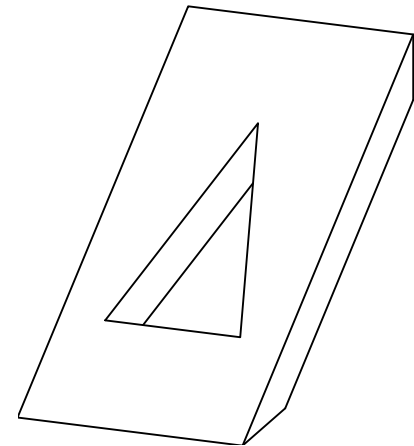
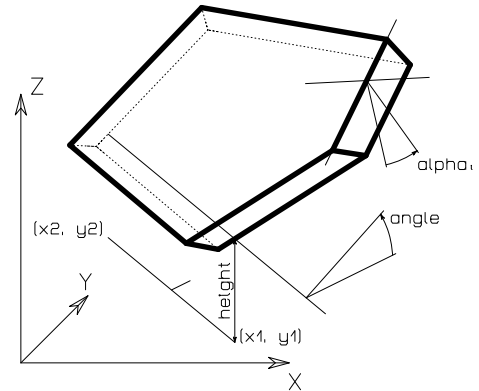
Ограничения на параметры:

$n \geq 3$

Примеры.

```

CROOF_ 1, 1, 1,           ! покрытия
      9,
      0, 0,
      1, 0,               ! базовая линия (xb, yb) (xe, ye)
      0.0,                ! высота
      -30,                ! угол поворота
      2.5,                ! толщина
      0, 0, -60, 15,
      10, 0, 0, 15,
      10, 20, -30, 15,
      0, 20, 0, 15,
      0, 0, 0, -1,
      2, 5, 0, 15,
      8, 5, 0, 15,
      5, 15, 0, 15,
      2, 5, 0, -1
    
```

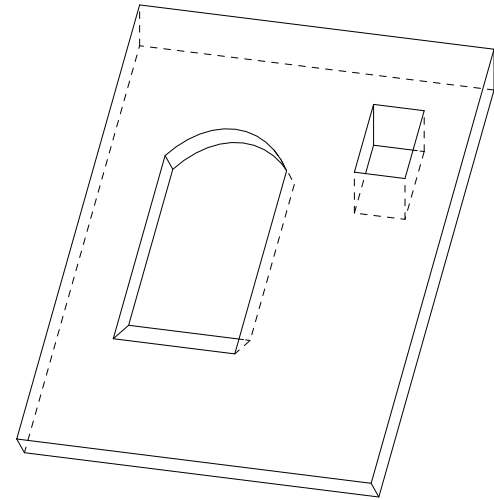




```

L=0.25
R=(0.6^2+L^2)/(2*L)
A=ASN(0.6/R)
CROOF_ "Черепица кровельная", "Сосна", "Сосна",
  16, 2, 0, 0,
  0, 0, 45, -0.2*SQR(2),
  0, 0, 0, 15,
  3.5, 0, 0, 15,
  3.5, 3, -45, 15,
  0, 3, 0, 15,
  0, 0, 0, -1,
  0.65, 1, -45, 15,
  1.85, 1, 0, 15,
  1.85, 2.4-L, 0, 13,
  1.25, 2.4-R, 0, 900,
  0, 2*A, 0, 4015,
  0.65, 1, 0, -1,
  2.5, 2, 45, 15,
  3, 2, 0, 15,
  3, 2.5, -45, 15,
  2.5, 2.5, 0, 15,
  2.5, 2, 0, -1

```



## MESH

**MESH**  $a, b, m, n, \text{mask},$   
 $z_{11}, z_{12}, \dots z_{1m},$   
 $z_{21}, z_{22}, \dots z_{2m},$   
 $\dots$   
 $z_{n1}, z_{n2}, \dots z_{nm}$

Простая сглаженная 3D-сетка на основе прямоугольника с равноотстоящей сетью. Сторонами прямоугольника являются  $a$  и  $b$ . Величины  $n$  и  $m$  указывают на количество равноотстоящих точек вдоль осей  $x$  и  $y$ , соответственно. Значением  $z_{ij}$  является высота соответствующего узла сети.

*Маскирование:*

$$\text{mask} = j_1 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7$$

где  $j_1, j_3, j_5, j_6, j_7$  могут принимать значения 0 или 1.

$j_1$  (1): поверхность основания присутствует.

$j_3$  (4): боковые поверхности присутствуют.

$j_5$  (16): ребра основания и боковых поверхностей присутствуют.

$j_6$  (32): верхние ребра присутствуют,

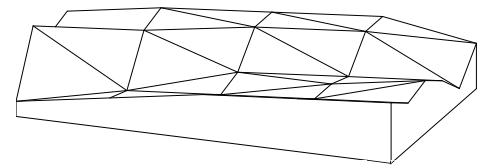
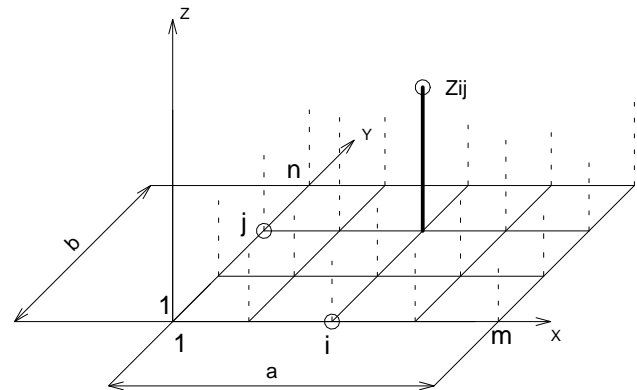
$j_7$  (64): ребра верхней поверхности присутствуют, верхняя поверхность не является сглаженной.

*Ограничения на параметры:*

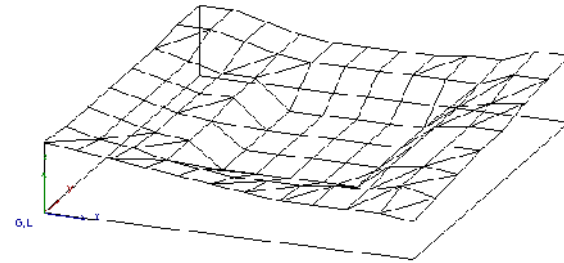
$$m \geq 2, \quad n \geq 2$$

*Примеры.*

MESH 50, 30, 5, 6, 1+4+16+32+64,  
 2, 4, 6, 7, 8,  
 10, 3, 4, 5, 6,  
 7, 9, 5, 5, 7,  
 8, 10, 9, 4, 5,  
 6, 7, 9, 8, 2,  
 4, 5, 6, 8, 6



```
MESH 90,100, 12,8, 1+4+16+32+64,
 17,16,15,14,13,12,11,10,10,10,10, 9,
 16,14,13,11,10, 9, 9, 9,10,10,12,10,
 16,14,12,11, 5, 5, 5, 5, 5,11,12,11,
 16,14,12,11, 5, 5, 5, 5, 5,11,12,12,
 16,14,12,12, 5, 5, 5, 5, 5,11,12,12,
 16,14,12,12, 5, 5, 5, 5, 5,11,13,14,
 17,17,15,13,12,12,12,12,12,12,15,15,
 17,17,15,13,12,12,12,12,13,13,16,16
```



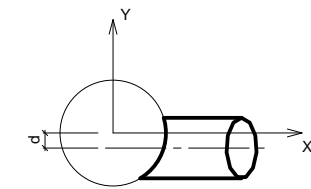
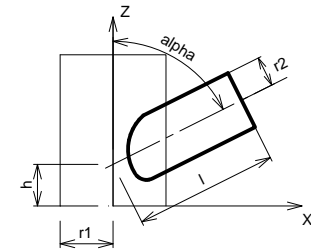
## ARMC

**ARMC** r1, r2, l, h, d, alpha

Часть цилиндра, выходящая из другого цилиндра. Смысл параметров иллюстрируется на рис. ниже. Линия сопряжения цилиндров также вычисляется и представляется. Угол alpha измеряется в градусах.

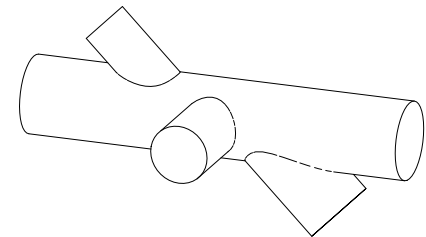
*Ограничения на параметры:*

$r1 \geq r2 + d$   
 $r1 \leq l \cdot \sin(\alpha) - r2 \cdot \cos(\alpha)$



*Пример:*

```
ROTY 90
CYLIND 10,1
ADDZ 6
ARMC 1, 0.9, 3, 0, 0, 45
ADDZ -1
ROTZ -90
ARMC 1, 0.75, 3, 0, 0, 90
ADDZ -1
ROTZ -90
ARMC 1, 0.6, 3, 0, 0, 135
```



## ARME

**ARME** l, r1, r2, h, d

Часть цилиндра, выходящая из эллипсоида в плоскости y-z. Смысл параметров иллюстрируется на рис. ниже. Линия сопряжения также вычисляется и представляется.

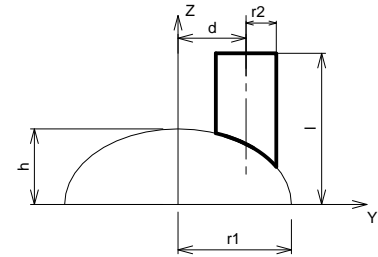
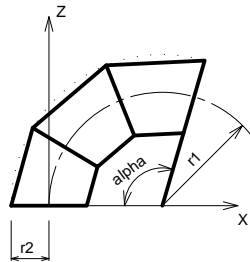
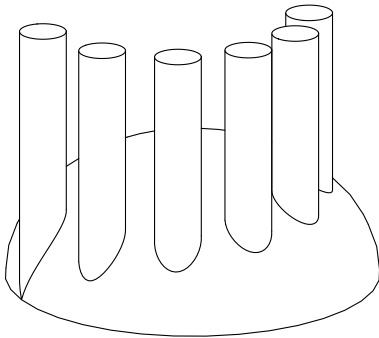
*Ограничения на параметры:*

$$r1 \geq r2 + d$$

$$l \geq h \cdot \sqrt{1 - (r2 - d)^2 / r1^2}$$

*Пример:*

```
ELLIPS 3,4
FOR i=1 TO 6
  ARME 6,4,0.5,3,3.7-0.2*i
  ROTZ 30
NEXT i
```



## ELBOW

**ELBOW** r1, alpha, r2

Изогнутый в плоскости x-z цилиндр (сегментированное колено). Радиус дуги равен r1, угол равен alpha, а радиус цилиндрических сегментов равен r2. Угол alpha измеряется в градусах.

*Ограничения на параметры:*

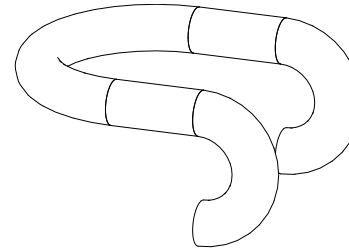
$$r1 > r2$$

Пример:

```

ROTY      90
ELBOW    2.5, 180, 1
ADDZ     -4
CYLIND   4, 1
ROTZ     -90
MULZ     -1
ELBOW    5, 180, 1
DEL      1
ADDX     10
CYLIND   4, 1
ADDZ     4
ROTZ     90
ELBOW    2.5, 180,1

```



## ПЛОСКИЕ ФИГУРЫ В ТРЕХМЕРНОМ ПРОСТРАНСТВЕ

Чертежные элементы, представленные в этом разделе, могут использоваться в 3D-скриптах для определения точек, линий, дуг, окружностей и плоских фигур в трехмерном пространстве.

### HOTSPOT

**HOTSPOT** *x, y, z* [, unID [, paramReference, flags] [, displayParam]]

Узловая точка в 3D с координатами (*x, y, z*).

unID: уникальный идентификатор узловой точки в 3D-скрипте. Оказывается полезным, если имеется переменное количество узловых точек.

paramReference: параметр, который может редактироваться с помощью этой узловой точки с использованием графического метода редактирования параметров в узловой точке.

displayParam: параметр, который показывается в информационной панели при редактировании параметра paramReference. Могут также передаваться элементы массивов.

См. [“Графическое редактирование” на стр. 135](#) для ознакомления со способами использованием предложения **HOTSPOT**.

### LIN\_

**LIN\_** *x1, y1, z1, x2, y2, z2*

Отрезок, ограниченный точками P1 (*x1,y1,z1*) и P2 (*x2,y2,z2*).

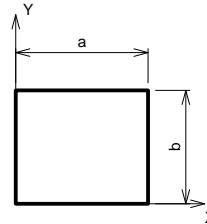
## RECT

**RECT** a, b

Прямоугольник в плоскости x-y со сторонами a и b.

*Ограничения на параметры:*

a, b  $\geq$  0



## POLY

**POLY** n, x1, y1, ... xn, yn

Многоугольник с n вершинами в плоскости. Координатами i-ой вершины являются (xi, yi, 0).

*Ограничения на параметры:*

n  $\geq$  3

## POLY\_

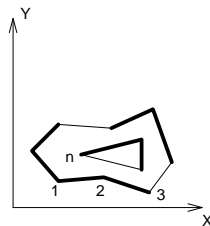
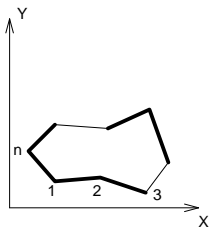
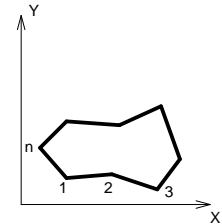
**POLY\_** n, x1, y1, s1, ... xn, yn, sn

Аналогично предложению POLY за исключением того, что могут быть опущены любые ребра. Если si = 0, то ребро, выходящее из вершины (xi, yi), отсутствует. Если si = 1, то соответствующее ребро присутствует.

Значение si = -1 используется для указания на то, что создается отверстие.

Дополнительные коды статуса позволяют создавать отрезки и дуги плоской ломаной линии с использованием специальных ограничений.

См. *“Дополнительные коды статусов” на стр. 143* для более детального ознакомления.



*Ограничения на параметры:*

n  $\geq$  3

## PLANE

**PLANE**  $n, x_1, y_1, z_1, \dots, x_n, y_n, z_n$

Многоугольник с  $n$  вершинами в произвольной плоскости. Координатами  $i$ -ой вершины являются  $(x_i, y_i, z_i)$ . Чтобы процессы отбрасывания теней и построения реалистических изображений были корректными, многоугольник должен быть плоским, однако интерпретатор не проверяет это условие.

*Ограничения на параметры:*

$n \geq 3$

## PLANE\_

**PLANE\_**  $n, x_1, y_1, z_1, s_1, \dots, x_n, y_n, z_n, s_n$

Аналогично предложению PLANE за исключением того, что, как и в POLY\_, могут быть опущены любые ребра.

Дополнительные коды статуса позволяют создавать отрезки и дуги плоской ломаной линии с использованием специальных ограничений.

См. [“Дополнительные коды статусов” на стр. 143](#).

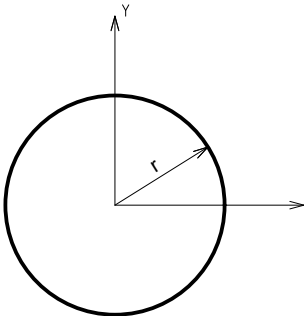
*Ограничения на параметры:*

$n \geq 3$

## CIRCLE

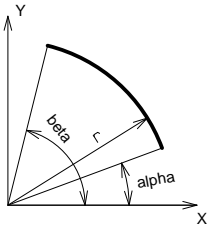
**CIRCLE**  $r$

Окружность в плоскости  $x$ - $y$  с центром в начале координат и радиусом  $r$ .



## ARC

**ARC** *r*, *alpha*, *beta*



Дуга (при построении каркасной модели) или сектор (при построении других моделей) в плоскости *x-y* с центром в начале координат, ограниченная углами *alpha* и *beta*, и имеющая радиус *r*. *alpha* и *beta* указываются в градусах.

## ФИГУРЫ, СОЗДАВАЕМЫЕ ИЗ ЛОМАННЫХ ЛИНИЙ

Эти элементы позволяют строить сложные 3D-фигуры на основе ломаных линий и встроенных правил. Вы можете поворачивать, проецировать или преобразовывать заданную ломаную линию. Создаваемые тела являются обобщением некоторых уже определенных элементов, например, `PRISM_` и `CYLIND`.

*Фигуры, создаваемые из одной ломаной линии:*

EXTRUDE  
PYRAMID  
REVOLVE

*Фигуры, создаваемые из двух ломаных линий:*

RULED  
SWEEP  
TUBE  
TUBEA

Первая ломаная всегда находится в плоскости *x-y*. Ее вершины определяются двумя координатами и статусом (см. ниже). Вторая ломаная (для `RULED` и `SWEEP`) находится в пространстве. Ее вершины определяются тремя координатами.

*Фигура, создаваемая из четырех ломаных:*

COONS

*Фигура, создаваемая из любого количества ломаных:*

MASS



## Общие ограничения на ломаные

- Соседние вершины не должны совпадать (за исключением RULED).
- Ломаная не должна самопересекаться (не проверяется, однако при самопересечении процессы удаления невидимых линий и построения реалистических изображений будут некорректными).
- Ломаные могут быть открытыми и замкнутыми. В последнем случае первая вершина должна быть повторена в конце предложения.

## Маскирование

Значения маски используются для того, чтобы указать, должны ли присутствовать в пространственной фигуре соответствующие ребра и поверхности. Значения маски являются специфическими для каждого элемента, и смысл их значений можно найти при описании соответствующих предложений.

$$\text{mask} = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7$$

где  $j1, j2, j3, j4, j5, j6, j7$  могут принимать значения 0 или 1.

Значения  $j1, j2, j3, j4$  указывают, присутствуют (1) или отсутствуют (0) соответствующие поверхности.

Значения  $j5, j6, j7$  указывают на присутствие, то есть являются ли они видимыми (1) или отсутствие, то есть являются ли они невидимыми (0), соответствующих ребер.

$j1$ : поверхность основания.

$j2$ : верхняя поверхность,.

$j3$ : боковая поверхность,.

$j4$ : другая боковая поверхность.

$j5$ : ребра основания.

$j6$ : ребра верхней поверхности.

$j7$ : ребра секущих плоскостей/поверхностей присутствуют, поверхность не является сглаженной.

Чтобы все ребра и поверхности присутствовали, установите значение маски равным 127.

## Статус

Значения статуса используются для того, чтобы указать будет ли соответствующая вершина ломаной оставлять отчетливый след при повороте.

0: все поперечные дуги/боковые ребра, исходящие из вершины, присутствуют.

1: поперечные дуги/боковые ребра, исходящие из вершин, используются только для показа контура.

-1: только для EXTRUDE: отмечает конец замкнутого многоугольника или отверстия и свидетельствует, что далее идет первая вершина следующего отверстия.

Дополнительные коды статуса позволяют создавать отрезки и дуги ломаной линии с использованием специальных ограничений. См. *“Дополнительные коды статусов” на стр. 143 для детального ознакомления.*

Для создания сглаженной 3D-фигуры установите значения всех значений статуса равными 1. Используйте статус 0 для получения ребра, исходящего из соответствующей вершины.

Другие значения зарезервированы для будущего использования.

## EXTRUDE

**EXTRUDE** *n*, *dx*, *dy*, *dz*, *mask*, *x1*, *y1*, *s1*,  
 ..., *xn*, *yn*, *sn*

Призма общего вида. Ломаная в плоскости x-y является основанием. Вектор смещения верхней поверхности относительно нижней равен (*dx*, *dy*, *dz*).

Является обобщением предложений PRISM и SLAB. Ломаная в основании может быть открытой, а боковые ребра не обязательно являются перпендикулярными плоскости x-y. Ломаная в основании может содержать отверстия, как и в PRISM\_. Имеется возможность управлять присутствием ребер контура.

*n*: количество вершин ломаной.

*mask*: управляет присутствием нижней, верхней и боковой (если ломаная открыта) поверхностей.

*si*: либо статус боковых ребер, либо признак конца описания ломаной или отверстия. Вы также можете определить отрезки и дуги ломаной линии с использованием дополнительных кодов статуса.

*Ограничения на параметры:*

$n > 2$

*Маскирование:*

$mask = j1 + 2*j2 + 4*j3 + 16*j5 + 32*j6$

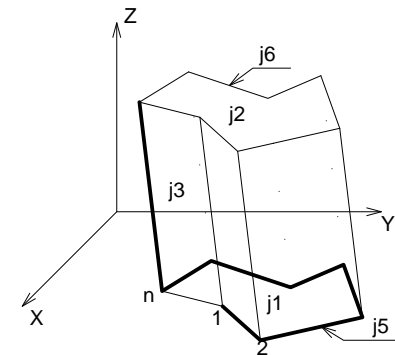
где *j1*, *j2*, *j3*, *j5*, *j6* могут принимать значения 0 или 1

*j1* (1): поверхность основания присутствует.

*j2* (2): верхняя поверхность присутствует.

*j3* (4): боковая грань (при разомкнутой ломаной) присутствует.

*j5* (16): ребра основания присутствуют.



j6 (32): ребра верхней поверхности присутствуют.

*Значения статуса:*

0: все боковые ребра, исходящие из вершин, присутствуют.

1: боковые ребра, исходящие из вершины, используются только для показа контура.

-1: отмечает конец замкнутого многоугольника или отверстия и свидетельствует, что далее идет первая вершина следующего отверстия.

Дополнительные коды статуса позволяют создавать отрезки и дуги плоской ломаной линии с использованием специальных ограничений.

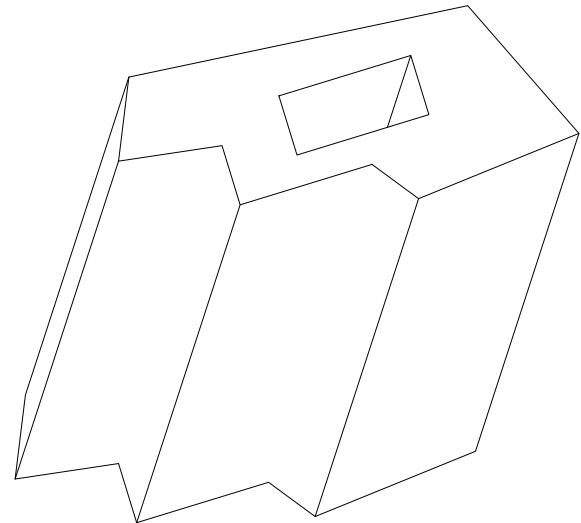
См. *“Дополнительные коды статусов”* на стр. 143 для детального ознакомления.

*Примеры.*

```

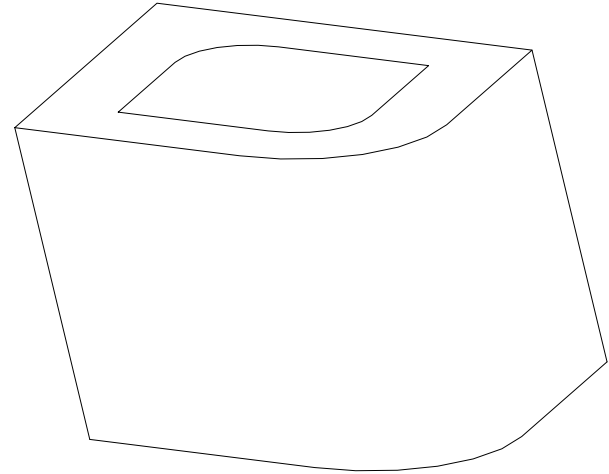
EXTRUDE 14, 1, 1, 4, 1+2+4+16+32,
0, 0, 0,
1, -3, 0,
2, -2, 1,
3, -4, 0,
4, -2, 1,
5, -3, 0,
6, 0, 0,
3, 4, 0,
0, 0, -1,
2, 0, 0,
3, 2, 0,
4, 0, 0,
3, -2, 0,
2, 0, -1

```



```

A=5: B=5
R=2: S=1
C=R-S
D=A-R
E=B-R
EXTRUDE 28, -1, 0, 4, 1+2+4+16+32,
0, 0, 0,
D+R*SIN(0), R-R*COS(0), 1,
D+R*SIN(15), R-R*COS(15), 1,
D+R*SIN(30), R-R*COS(30), 1,
D+R*SIN(45), R-R*COS(45), 1,
D+R*SIN(60), R-R*COS(60), 1,
D+R*SIN(75), R-R*COS(75), 1,
D+R*SIN(90), R-R*COS(90), 1,
A, B, 0,
0, B, 0,
0, 0, -1,
C, C, 0,
D+S*SIN(0), R-S*COS(0), 1,
D+S*SIN(15), R-S*COS(15), 1,
D+S*SIN(30), R-S*COS(30), 1,
D+S*SIN(45), R-S*COS(45), 1,
D+S*SIN(60), R-S*COS(60), 1,
D+S*SIN(75), R-S*COS(75), 1,
D+S*SIN(90), R-S*COS(90), 1,
A-C, B-C, 0,
R-S*COS(90), E+S*SIN(90), 1,
R-S*COS(75), E+S*SIN(75), 1,
R-S*COS(60), E+S*SIN(60), 1,
R-S*COS(45), E+S*SIN(45), 1,
R-S*COS(30), E+S*SIN(30), 1,
R-S*COS(15), E+S*SIN(15), 1,
R-S*COS(0), E+S*SIN(0), 1,
C, C, -1
    
```



## PYRAMID

**PYRAMID** *n*, *h*, *mask*, *x1*, *y1*, *s1*, ... *xn*, *yn*, *sn*

Пирамида. Ломаная в основании расположена в плоскости *x-y*. Вершина пирамиды расположена в точке с координатами (0, 0, *h*).

*n*: количество вершин ломаной.

*mask*: управляет присутствием нижней и боковой (если ломаная открыта) поверхностей.

*si*: статус боковых ребер.

*Ограничения на параметры:*

$h > 0$  и  $n > 2$

*Маскирование:*

$mask = j1 + 4*j3 + 16*j5$

где *j1*, *j3*, *j5* могут принимать значения 0 или 1

*j1* (1): поверхность основания присутствует.

*j3* (4): боковая грань (при разомкнутой ломаной) присутствует.

*j5* (16): ребра основания присутствуют.

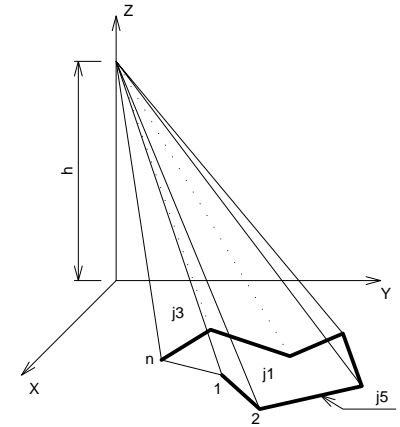
*Значения статуса:*

0: все боковые ребра, исходящие из вершин, присутствуют.

1: боковые ребра, исходящие из вершин, используется только для показа контура.

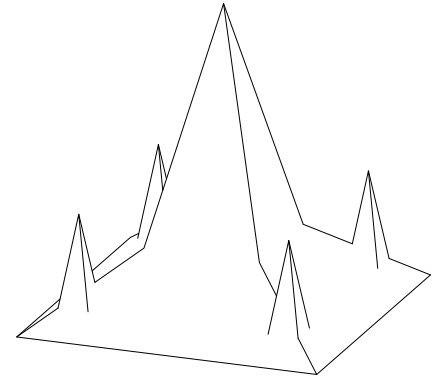
Дополнительные коды статуса позволяют создавать отрезки и дуги плоской ломаной линии с использованием специальных ограничений.

См. "*Дополнительные коды статусов*" на стр. 143 для детального ознакомления.



*Пример:*

```
PYRAMID 4, 1.5, 1+4+16,  
        -2, -2, 0,  
        -2, 2, 0,  
        2, 2, 0,  
        2, -2, 0  
PYRAMID 4, 4, 21,  
        -1, -1, 0,  
        1, -1, 0,  
        1, 1, 0,  
        -1, 1, 0  
ADDX -1.4  
ADDY -1.4  
GOSUB 100  
ADDX 2.8  
GOSUB 100  
ADDY 2.8  
GOSUB 100  
ADDX -2.8  
GOSUB 100  
END  
100:  
PYRAMID 4, 1.5, 21,  
        -0.25, -0.25, 0,  
        0.25, -0.25, 0,  
        0.25, 0.25, 0,  
        -0.25, 0.25, 0  
RETURN
```



## REVOLVE

**REVOLVE** *n*, *alpha*, *mask*, *x1*, *y1*, *s1*, ... *xn*, *yn*, *sn*

Поверхность поворота ломаной в плоскости x-y вокруг оси x.

*n*: количество вершин ломаной.

*alpha*: угол поворота в градусах.

*mask*: управляет присутствием нижней, верхней и боковых (если  $\alpha < 360^\circ$ ) поверхностей.

*si*: статус поперечных дуг.

*Ограничения на параметры:*

$n \geq 2$

$y_i \geq 0.0$

$y_i$  и  $y_{i+1}$  (то есть значения  $y$  двух соседних вершин) не могут быть одновременно равны 0.

*Маскирование:*

$mask = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7$

где  $j1, j2, j3, j4, j5, j6, j7$  могут принимать значения 0 или 1

$j1$  (1): поверхность основания присутствует.

$j2$  (2): верхняя поверхность присутствует.

$j3$  (4): боковая поверхность при начальной стороне угла присутствует.

$j4$  (8): боковая поверхность при конечной стороне угла присутствует.

$j5$  (16): ребра боковой поверхности при начальной стороне угла присутствуют.

$j6$  (32): ребра боковой поверхности при конечной стороне угла присутствуют.

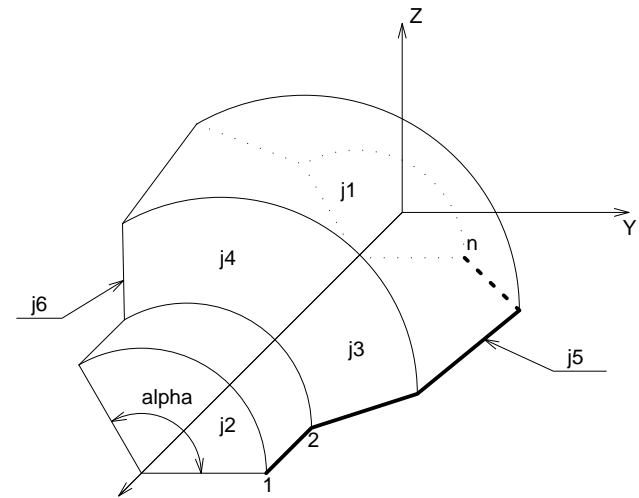
$j7$  (64): ребра осевых секущих плоскостей присутствуют, поверхность не является сглаженной.

*Значения статуса:*

0: все поперечные ребра, исходящие из вершин, присутствуют.

1: поперечные ребра, исходящие из вершин, используются только для показа контура.

2: При использовании механизмов визуализации ArchiCAD/ArchiFM или Z-buffer и установке сглаживания поверхностей, поперечные ребра, проходящие через эту точку, определяют линию излома. Данное решение



эквивалентно определению дополнительных вершин. Это осуществляется компилятором. При использовании других методов построения реалистических изображений эффект будет аналогичен случаю, когда значение статуса 0.

Дополнительные коды статуса позволяют создавать отрезки и дуги плоской ломаной линии с использованием специальных ограничений.

См. *“Дополнительные коды статусов” на стр. 143* для детального ознакомления.

**REVOLVE{2}** *n*, alphaOffset, alpha, mask, sideMat, x1, y1, s1, mat1, ...xn, yn, sn, matn

Усовершенствованная форма REVOLVE, где управляется задание начального угла и покрытий поверхностей.

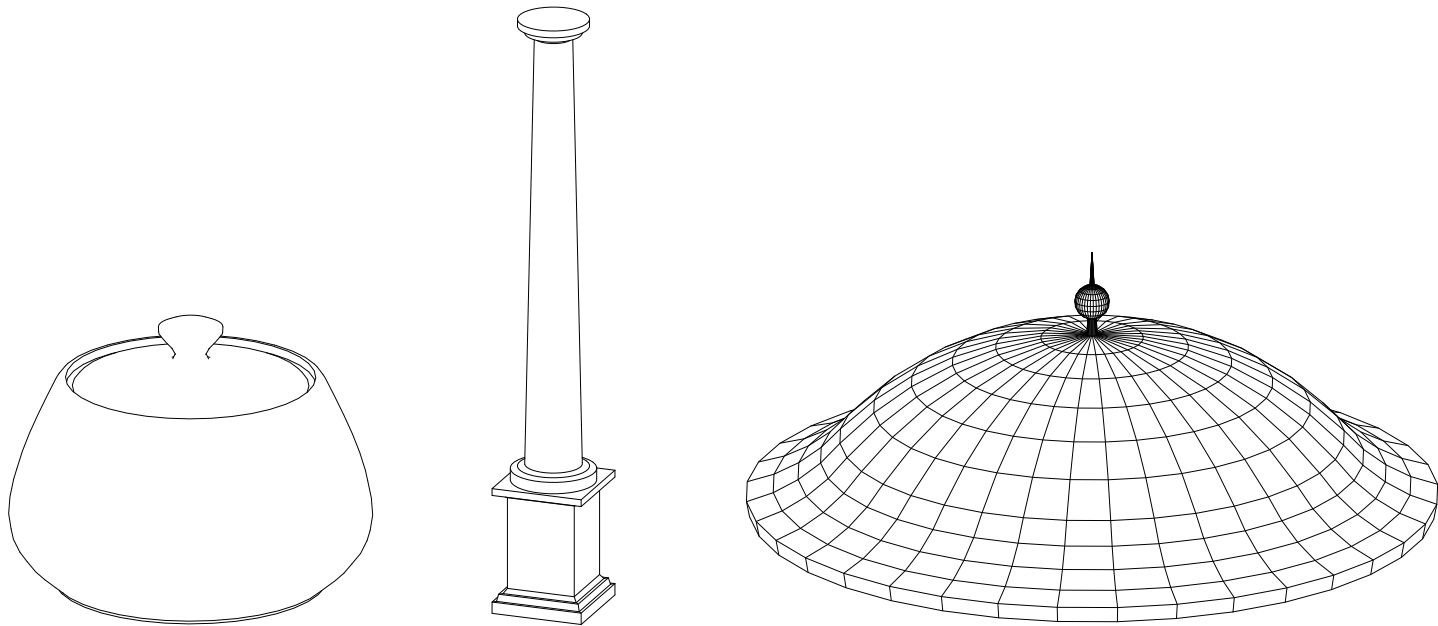
alphaOffset: начальный угол поворота.

alpha: длина угла поворота, может быть отрицательной.

sideMat: покрытие замыкающих поверхностей.

mat<sub>i</sub>: покрытие поверхности, создаваемой из *i*-го ребра.

*Примеры:*





```
ROTY -90
REVOLVE 22, 360, 1+64,
0, 1.982, 0,
0.093, 2, 0,
0.144, 1.845, 0,
0.220, 1.701, 0,
0.318, 1.571, 0,
0.436, 1.459, 0,
0.617, 1.263, 0,
0.772, 1.045, 0,
0.896, 0.808, 0,
0.987, 0.557, 0,
1.044, 0.296, 0,
1.064, 0.030, 0,
1.167, 0.024, 0,
1.181, 0.056, 0,
1.205, 0.081, 0,
1.236, 0.096, 0,
1.270, 0.1, 0,
1.304, 0.092, 0,
1.333, 0.073, 0,
1.354, 0.045, 0,
1.364, 0.012, 0,
1.564, 0, 0
```

**Без  
кода статуса 2:**

```

ROTY -90
REVOLVE 26, 180, 16+32,
7, 1, 0,
6.0001, 1, 1,
6, 1, 0,
5.9999, 1.0002, 1,
5.5001, 1.9998, 1,
5.5, 2, 0,
5.4999, 1.9998, 1,
5.0001, 1.0002, 1,
5, 1, 0,
4.9999, 1, 1,
4.0001, 1, 1,
4, 1, 0,
3+COS(15), 1+SIN(15), 1,
3+COS(30), 1+SIN(30), 1,
3+COS(45), 1+SIN(45), 1,
3+COS(60), 1+SIN(60), 1,
3+COS(75), 1+SIN(75), 1,
3, 2, 1,
3+COS(105), 1+SIN(105), 1,
3+COS(120), 1+SIN(120), 1,
3+COS(135), 1+SIN(135), 1,
3+COS(150), 1+SIN(150), 1,
3+COS(165), 1+SIN(165), 1,
2, 1, 0,
1.9999, 1, 0,
1, 1, 0
    
```

**RULED**

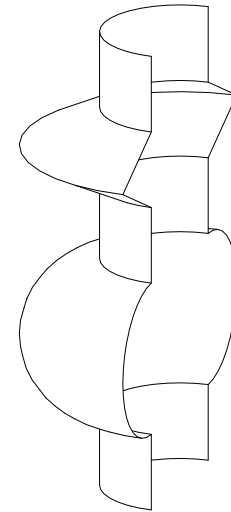
```

RULED n, mask,
    u1, v1, s1, ... un, vn, sn,
    x1, y1, z1, ... xn, yn, zn
    
```

**С кодом  
статуса 2:**

```

ROTY -90
REVOLVE 18, 180, 48,
7, 1, 0,
6, 1, 2,
5.5, 2, 2,
5, 1, 2,
4, 1, 2,
3+COS(15), 1+SIN(15), 1,
3+COS(30), 1+SIN(30), 1,
3+COS(45), 1+SIN(45), 1,
3+COS(60), 1+SIN(60), 1,
3+COS(75), 1+SIN(75), 1,
3, 2, 1,
3+COS(105), 1+SIN(105), 1,
3+COS(120), 1+SIN(120), 1,
3+COS(135), 1+SIN(135), 1,
3+COS(150), 1+SIN(150), 1,
3+COS(165), 1+SIN(165), 1,
2, 1, 2,
1, 1, 0
    
```



## RULED{2}

**RULED{2}**  $n$ ,  $mask$ ,  
 $u_1, v_1, s_1, \dots, u_n, v_n, s_n$ ,  
 $x_1, y_1, z_1, \dots, x_n, y_n, z_n$

RULED - это поверхность, созданная на основе ломаной в плоскости и ломаной в пространстве, имеющих одинаковое количество вершин. Соответствующие вершины ломаных соединяются прямолинейными отрезками.

Это единственная фигура GDL, допускающая совпадение соседних вершин ломаной. Второй вариант этого предложения, RULED{2}, проверяет направление (по часовой стрелке или против), в котором определяются точки многоугольников вверху и внизу, и при необходимости изменяет направление. (Исходная команда RULED принимает в расчет только многоугольник в основании, что может приводить к ошибкам.)

$n$ : количество вершин в каждой ломаной.

$mask$ : управляет присутствием нижней, верхней и боковой поверхностей, а также ребер образующих ломаных. Боковая поверхность образуется многоугольником, соединяющим первые и последние вершины ломаных, если хотя бы одна из них не является замкнутой.

$u_i, v_i$ : координаты вершин ломаной линии в плоскости.

$s_i$ : статус боковых ребер.

$x_i, y_i, z_i$ : координаты вершин ломаной линии в пространстве.

*Ограничения на параметры:*

$n > 1$

*Маскирование:*

$mask = j_1 + 2*j_2 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7$

где  $j_1, j_2, j_3, j_5, j_6, j_7$  могут принимать значения 0 или 1.

$j_1$  (1): поверхность основания присутствует.

$j_2$  (2): верхняя поверхность присутствует (не используется, если она не плоская).

$j_3$  (4): боковая поверхность (при разомкнутом звене ломаной) присутствует (четырёхугольник или два треугольника).

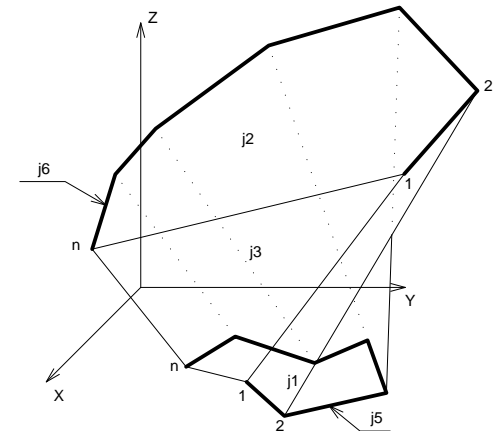
$j_5$  (16): ребра ломаной в плоскости присутствуют.

$j_6$  (32): ребра ломаной в пространстве присутствуют.

$j_7$  (64): ребра боковой поверхности присутствуют, поверхность не является сглаженной.

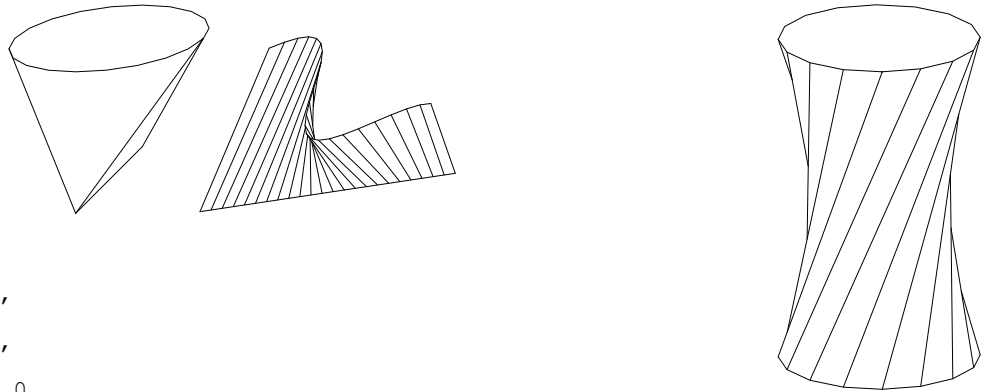
*Значения статуса:*

0: все боковые ребра, исходящие из вершин, присутствуют.



1: боковые ребра, исходящие из вершин, используется только для показа контура.

Примеры:



```

R=3
RULED 16, 1+2+4+16+32,
COS(22.5)*R, SIN(22.5)*R, 0,
COS(45)*R, SIN(45)*R, 0,
COS(67.5)*R, SIN(67.5)*R, 0,
COS(90)*R, SIN(90)*R, 0,
COS(112.5)*R, SIN(112.5)*R, 0,
COS(135)*R, SIN(135)*R, 0,
COS(157.5)*R, SIN(157.5)*R, 0,
COS(180)*R, SIN(180)*R, 0,
COS(202.5)*R, SIN(202.5)*R, 0,
COS(225)*R, SIN(225)*R, 0,
COS(247.5)*R, SIN(247.5)*R, 0,
COS(270)*R, SIN(270)*R, 0,
COS(292.5)*R, SIN(292.5)*R, 0,
COS(315)*R, SIN(315)*R, 0,
COS(337.5)*R, SIN(337.5)*R, 0,
COS(360)*R, SIN(360)*R, 0,
COS(112.5)*R, SIN(112.5)*R, 1,
COS(135)*R, SIN(135)*R, 1,
COS(157.5)*R, SIN(157.5)*R, 1,
COS(180)*R, SIN(180)*R, 1,
COS(202.5)*R, SIN(202.5)*R, 1,
COS(225)*R, SIN(225)*R, 1,
COS(247.5)*R, SIN(247.5)*R, 1,
COS(270)*R, SIN(270)*R, 1,
COS(292.5)*R, SIN(292.5)*R, 1,
COS(315)*R, SIN(315)*R, 1,
COS(337.5)*R, SIN(337.5)*R, 1,
COS(360)*R, SIN(360)*R, 1,
COS(22.5)*R, SIN(22.5)*R, 1,
COS(45)*R, SIN(45)*R, 1,
COS(67.5)*R, SIN(67.5)*R, 1,
COS(90)*R, SIN(90)*R, 1
    
```

## **SWEEP**

**SWEEP**  $n, m, \alpha, \text{scale}, \text{mask},$   
 $u_1, v_1, s_1, \dots, u_n, v_n, s_n,$   
 $x_1, y_1, z_1, \dots, x_m, y_m, z_m$

Фигура, созданная перемещением образующей ломаной по направляющей ломаной в пространстве.

Плоскость образующей ломаной следует по направляющей ломаной. Направляющая должна начинаться в плоскости  $x$ - $y$ . Если это условие не выполняется, то ломаная перемещается вдоль оси  $z$  до соприкосновения с плоскостью  $x$ - $y$ .

Плоскость образующей ломаной в точке  $(x_i, y_i, z_i)$  перпендикулярна отрезку направляющей ломаной между точками  $(x_{i-1}, y_{i-1}, z_{i-1})$  и  $(x_i, y_i, z_i)$ .

SWEEP может использоваться для создания таких довольно сложных фигур, как, например, носик чайника.

$n$ : количество вершин образующей ломаной.

$m$ : количество вершин направляющей ломаной.

$\alpha$ : угол поворота образующей ломаной в своей плоскости при переходе от одной вершины направляющей к другой.

$\text{scale}$ : масштабный множитель образующей ломаной при переходе от одной вершины направляющей ломаной к другой.

$\text{mask}$ : управляет присутствием поверхностей при начальном и конечном состоянии образующей ломаной, а также ее ребер.

$u_i, v_i$ : координаты вершин ломаной в основании.

$s_i$ : статус боковых ребер.

$x_i, y_i, z_i$ : координаты вершин направляющей ломаной.

*Ограничения на параметры:*

$n > 1$   
 $m > 1$   
 $z_1 < z_2$

*Маскирование:*

$$\text{mask} = j1 + 2*j2 + 4*j3 + 16*j5 + 32*j6 + 64*j7$$

где  $j1, j2, j3, j5, j6, j7$  могут принимать значения 0 или 1

$j1$  (1): поверхность основания присутствует.

$j2$  (2): конечная поверхность присутствует.

$j3$  (4): боковая поверхность при разомкнутом звене ломаной присутствует.

$j5$  (16): ребра основания присутствуют.

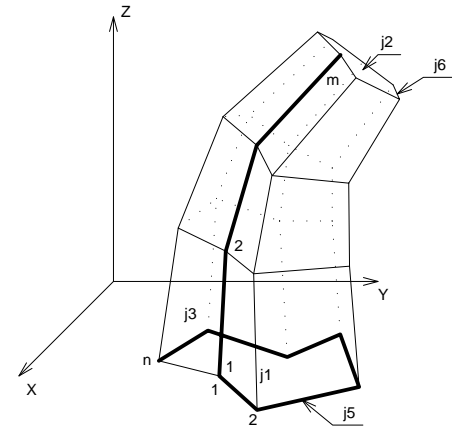
$j6$  (32): ребра верхней поверхности присутствуют.

$j7$  (64): ребра в плоскостях образующей ломаной присутствуют, поверхность представлена в виде сочленений.

*Значения статуса:*

0: все боковые ребра, исходящие из вершин, присутствуют.

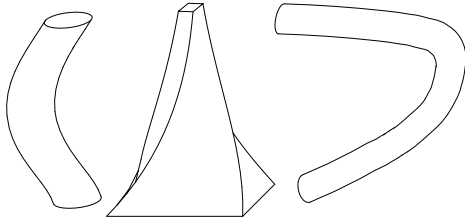
1: боковые ребра, исходящие из вершин, используется только для показа контура.



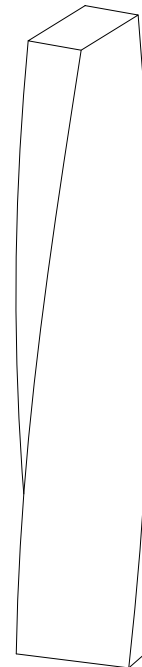
Дополнительные коды статуса позволяют создавать отрезки и дуги плоской ломаной линии с использованием специальных ограничений.

См. *“Дополнительные коды статусов”* на стр. 143 для детального ознакомления.

Примеры:



```
SWEEP 4, 12, 7.5, 1, 1+2+4+16+32,
-0.5, -0.25, 0,
0.5, -0.25, 0,
0.5, 0.25, 0,
-0.5, 0.25, 0,
0, 0, 0.5,
0, 0, 1,
0, 0, 1.5,
0, 0, 2,
0, 0, 2.5,
0, 0, 3,
0, 0, 3.5,
0, 0, 4,
0, 0, 4.5,
0, 0, 5,
0, 0, 5.5,
0, 0, 6
```



## TUBE

**TUBE** *n*, *m*, *mask*,  
*u1*, *w1*, *s1*,  
 ...  
*un*, *wn*, *sn*,  
*x1*, *y1*, *z1*, *angle1*,  
 ...  
*xm*, *ym*, *zm*, *anglem*

Фигура, созданная перемещением образующей ломаной по направляющей ломаной в пространстве без искажения образующей плоскости. Образующая ломаная располагается в плоскости U-W изменяющейся системы координат U-V-W.

Ось V: аппроксимирует касательную к направляющей в соответствующей вершине.

Ось W: перпендикулярна оси V и указывает в сторону направления локальной оси z.

Ось U: перпендикулярна осям V и W и образует с ними правую декартову систему координат.

Если ось V вертикальна, то направление W определяется некорректно. Ось W предыдущей вершины направляющей используется для определения горизонтального направления.

Многоугольник образующей плоскости, создаваемый в середине отрезка направляющей ломаной, всегда равен многоугольнику в основании (*u1*, *w1*, ... *un*, *wn*). Многоугольник образующей плоскости, проходящий через вершину направляющей ломаной, располагается в плоскости равноудаленной от соседних отрезков направляющей ломаной (то есть эта плоскость является биссектрисой угла, определяемого вершиной и двумя соседними отрезками направляющей ломаной). Многоугольник в основании должен быть замкнутым.

*n*: количество вершин образующей ломаной.

*m*: количество вершин направляющей ломаной.

*u<sub>i</sub>*, *w<sub>i</sub>*: координаты вершин образующей ломаной в основании.

*s<sub>i</sub>*: статус боковых ребер.

*x<sub>i</sub>*, *y<sub>i</sub>*, *z<sub>i</sub>*: координаты вершин направляющей ломаной.

**Примечание:** Направляющая ломаная содержит на две точки больше, чем число образующих плоскостей. Начальная и конечная точки направляющей определяют расположение в пространстве первой и последней поверхностей, принадлежащих TUBE. Эти точки нужны только для того, чтобы определить нормали этих двух поверхностей, и они фактически не являются вершинами направляющей ломаной. Ориентация этих поверхностей совпадает с ориентацией поверхностей, которые могли бы быть: построены в вершинах, ближайших к этим двум конечным точкам, если бы TUBE был продолжен в указанных ими направлениях.

*angle<sub>i</sub>*: угол поворота плоскости образующей ломаной.



*Маскирование:*

$mask = j1 + 2*j2 + 16*j5 + 32*j6 + 64*j7$

где  $j1, j2, j5, j6, j7$  могут принимать значения 0 или 1

$j1$  (1): поверхность основания присутствует.

$j2$  (2): конечная поверхность присутствует.

$j5$  (16): ребра поверхности основания (в  $x2, y2, z2$ ) присутствуют.

$j6$  (32): ребра конечной поверхности (в  $xm-1, ym-1, zm-1$ ) присутствуют.

$j7$  (64): ребра в плоскостях образующей ломаной присутствуют, поверхность представлена в виде сочленений.

*Ограничения на параметры:*

$n > 2$  и  $m > 3$

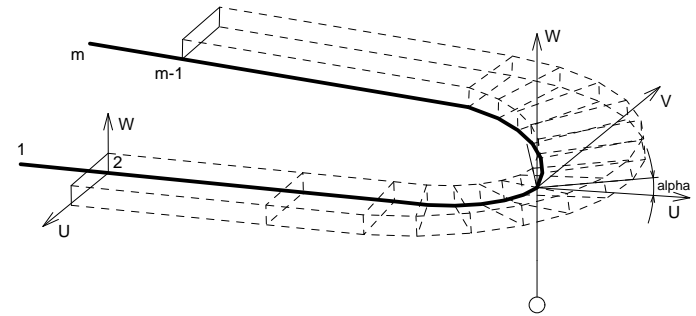
*Значения статуса:*

0: все боковые ребра, исходящие из вершин, присутствуют.

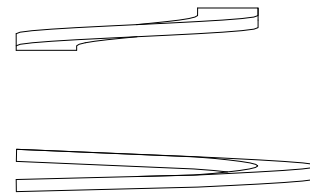
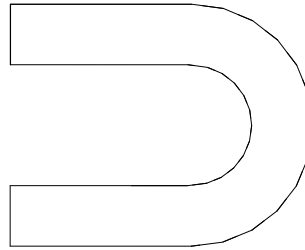
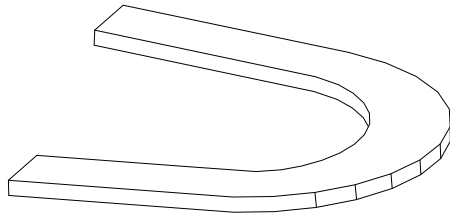
1: боковые ребра, исходящие из вершин, используется только для показа контура.

Дополнительные коды статуса позволяют создавать отрезки и дуги плоской ломаной линии с использованием специальных ограничений.

См. "Параметрические объекты" в главе "Виртуальное здание" справки ArchiCAD 11.



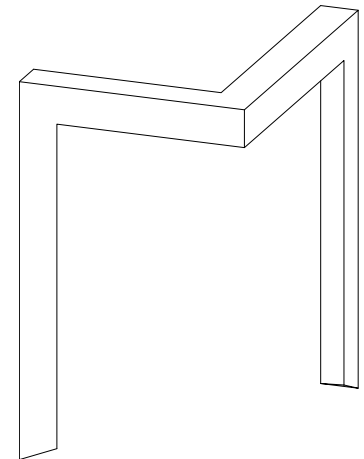
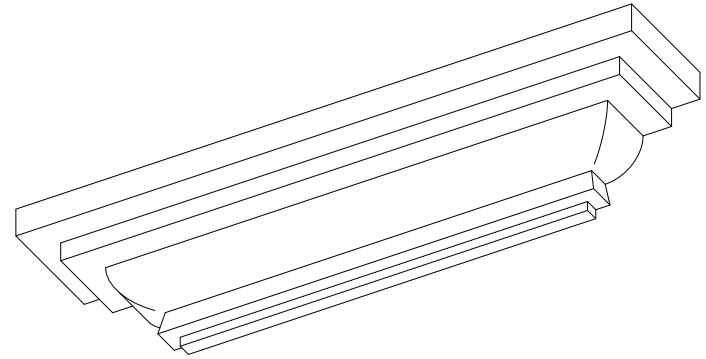
Примеры:



```
TUBE 4, 18, 16+32,
    2.0, 0.0, 0,
    0.0, 0.0, 0,
    0.0, 0.4, 0,
    2.0, 0.4, 0,
    -1, 0, 0, 0,
    0, 0, 0, 0,
    4, 0, 0.1, 0,
    6, 0, 0.15, 0,
    6+4*SIN(15), 4 - 4*COS(15), 0.2, 0,
    6+4*SIN(30), 4 - 4*COS(30), 0.25, 0,
    6+4*SIN(45), 4 - 4*COS(45), 0.3, 0,
    6+4*SIN(60), 4 - 4*COS(60), 0.35, 0,
    6+4*SIN(75), 4 - 4*COS(75), 0.4, 0,
    10, 4, 0.45, 0,
    6+4*SIN(105), 4 - 4*COS(105), 0.5, 0,
    6+4*SIN(120), 4 - 4*COS(120), 0.55, 0,
    6+4*SIN(135), 4 - 4*COS(135), 0.6, 0,
    6+4*SIN(150), 4 - 4*COS(150), 0.65, 0,
    6+4*SIN(165), 4 - 4*cos(165), 0.7, 0,
    6, 8, 0.75, 0,
    0, 8, 1, 0,
    -1, 8, 1, 0
```

```
TUBE 14, 6, 1+2+16+32,
0, 0, 0,
0.03, 0, 0,
0.03, 0.02, 0,
0.06, 0.02, 0,
0.05, 0.0699, 0,
0.05, 0.07, 1,
0.05, 0.15, 901,
1, 0, 801,
0.08, 90, 2000,
0.19, 0.15, 0,
0.19, 0.19, 0,
0.25, 0.19, 0,
0.25, 0.25, 0,
0, 0.25, 0,
0, 1, 0, 0,
0, 0.0001, 0, 0,
0, 0, 0, 0,
-0.8, 0, 0, 0,
-0.8, 0.0001, 0, 0,
-0.8, 1, 0, 0
```

```
TUBE 3, 7, 16+32,
0, 0, 0,
-0.5, 0, 0,
0, 0.5, 0,
0.2, 0, -0.2, 0,
0, 0, 0, 0,
0, 0, 5, 0,
3, 0, 5, 0,
3, 4, 5, 0,
3, 4, 0, 0,
3, 3.8, -0.2, 0
```



## TUBEA

**TUBEA**  $n, m, mask,$   
 $u1, w1, s1,$   
 $\dots$   
 $un, wn, sn,$   
 $x1, y1, z1,$   
 $\dots$   
 $xm, ym, zm$

TUBEA - это поверхность, созданная перемещением образующей ломаной по направляющей ломаной в пространстве, но алгоритм построения отличается от определенного в предложении TUBE.

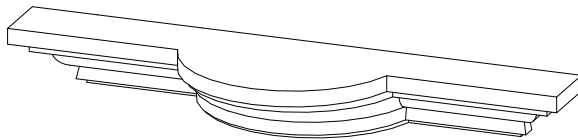
В каждой вершине направляющей ломаной строятся образующие многоугольники, равные многоугольнику в основании ( $u1, w1, \dots, un, wn$ ) и располагающиеся в плоскости, равноудаленной от проекций соседних отрезков направляющей ломаной на плоскость  $x-y$  локальной системы координат. Многоугольник в основании может быть не замкнутым. В этом случае для построения образующих многоугольников образующая ломаная продлевается до пересечения с плоскостью  $x-y$  локальной системы координат как в случае поверхностей REVOLVE.

Сечение, построенное в середине отрезка направляющей ломаной, может отличаться от многоугольника в основании.

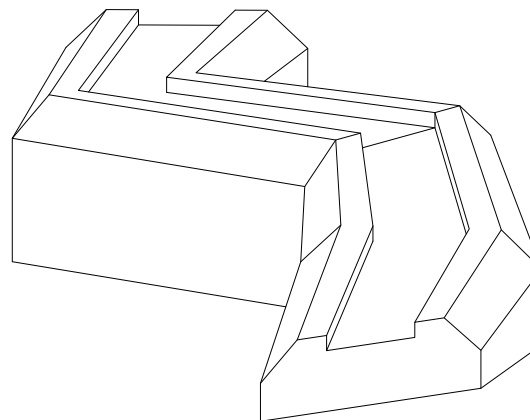
Дополнительные коды статуса позволяют создавать отрезки и дуги плоской ломаной линии с использованием специальных ограничений.

См. *“Дополнительные коды статусов”* на стр. 143 для детального ознакомления.

Примеры:



TUBEA 9, 7, 1 + 2 + 16 + 32,  
 -1, 1, 0,  
 0, 2, 0,  
 0.8, 2, 0,  
 0.8, 1.6, 0,  
 0.8001, 1.6, 1,  
 3.2, 1.6, 0,  
 3.2, 2, 0,  
 4, 2, 0,  
 5, 1, 0,  
 0, -7, 0,  
 0, 0, 0,  
 4, 0, 1,  
 9, 3, 2.25,  
 9, 10, 2.25,  
 14, 10, 2.25,  
 20, 15, 5



## COONS

**COONS**  $n, m, \text{mask},$   
 $x_{11}, y_{11}, z_{11}, \dots, x_{1n}, y_{1n}, z_{1n},$   
 $x_{21}, y_{21}, z_{21}, \dots, x_{2n}, y_{2n}, z_{2n},$   
 $x_{31}, y_{31}, z_{31}, \dots, x_{3m}, y_{3m}, z_{3m},$   
 $x_{41}, y_{41}, z_{41}, \dots, x_{4m}, y_{4m}, z_{4m}$

Фрагмент поверхности Кунса, строящийся четырьмя примыкающими кривыми.

*Маскирование:*

$\text{mask} = 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7$

где  $j_3, j_4, j_5, j_6, j_7$  могут принимать значения 0 или 1

$j_3$  (4): ребра 1-й границы  $(x_1, y_1, z_1)$  присутствуют.

$j_4$  (8): ребра 2-й границы  $(x_2, y_2, z_2)$  присутствуют.

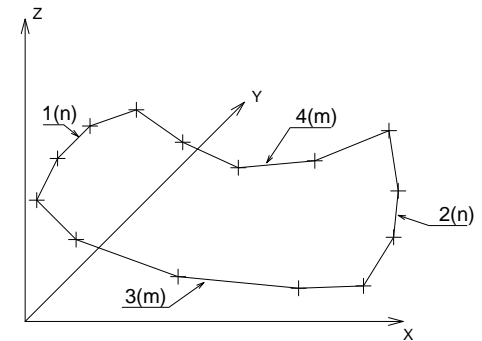
$j_5$  (16): ребра 3-й границы  $(x_3, y_3, z_3)$  присутствуют.

$j_6$  (32): ребра 4-й границы  $(x_4, y_4, z_4)$  присутствуют.

$j_7$  (64): ребра поверхности присутствуют, поверхность не является сглаженной.

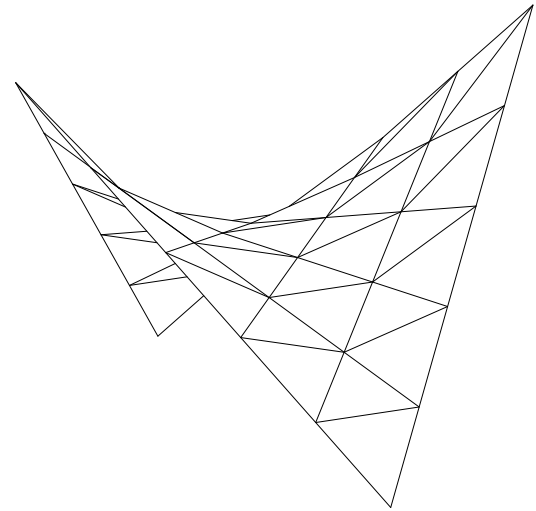
*Ограничения на параметры:*

$n, m > 1$



*Примеры:*

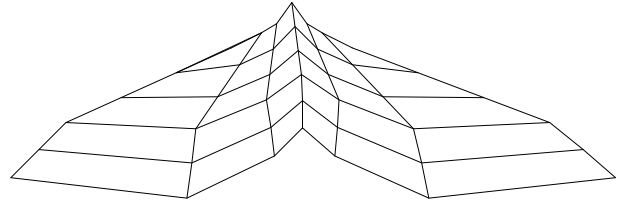
```
COONS  6, 6, 4+8+16+32+64,
! 1-я граница, n=6
0, 0, 5,
1, 0, 4,
2, 0, 3,
3, 0, 2,
4, 0, 1,
5, 0, 0,
! 2-я граница, n=6
0, 5, 0,
1, 5, 1,
2, 5, 2,
3, 5, 3,
4, 5, 4,
5, 5, 5,
! 3-я граница, m=6
0, 0, 5,
0, 1, 4,
0, 2, 3,
0, 3, 2,
0, 4, 1,
0, 5, 0,
! 4-я граница, m=6
5, 0, 0,
5, 1, 1,
5, 2, 2,
5, 3, 3,
5, 4, 4,
5, 5, 5
```



```

ROTZ  -90
ROTY  90
COONS 7, 6, 4+8+16+32+64,
! 1-я граница, n=7
1, 2, 0,
0.5, 1, 0,
0.2, 0.5, 0,
-0.5, 0, 0,
0.2, -0.5, 0,
0.5, -1, 0,
1, -2, 0,
! 2-я граница, n=7
6, 10, -2,
6.5, 4, -1.5,
5, 1, -1.2,
4, 0, -1,
5, -1, -1.2,
6.5, -4, -1.5,
6, -10, -2,
! 3-я граница, m=6
1, 2, 0,
2, 4, -0.5,
3, 6, -1,
4, 8, -1.5,
5, 9, -1.8,
6, 10, -2,
! 4-я граница, m=6
1, -2, 0,
2, -4, -0.5,
3, -6, -1,
4, -8, -1.5,
5, -9, -1.8,
6, -10, -2

```





## MASS

```

MASS top_material, bottom_material, side_material,
      n, m, mask, h,
      x1, y1, z1, s1,
      ...
      xn, yn, zn, sn,
      xn+1, yn+1, zn+1, sn+1,
      ...
      xn+m, yn+m, zn+m, sn+m
  
```

Это предложение эквивалентно созданию фигур с помощью инструмента 3D-сетка в ArchiCAD.

`top_material`, `bottom_material`, `side_material`: имя/индекс покрытий для верхней, нижней и боковых поверхностей.

`n`: количество вершин в многоугольнике тела.

`m`: количество вершин на ребрах.

`h`: высота боковых поверхностей (может быть отрицательна).

`xi`, `yi`, `zi`: координаты вершин.

`si`: аналогично предложению `PRISM_`. Дополнительные коды статуса позволяют создавать отрезки и дуги плоской ломаной линии с использованием специальных ограничений.

См. “Дополнительные коды статусов” на стр. 143 для детального ознакомления.

**Маскирование:**

```

mask = j1 + 4*j3 + 16*j5 + 32*j6 + 64*j7
  
```

где `j1`, `j3`, `j5`, `j6`, `j7` могут принимать значения 0 или 1

`j1` (1): поверхность основания присутствует.

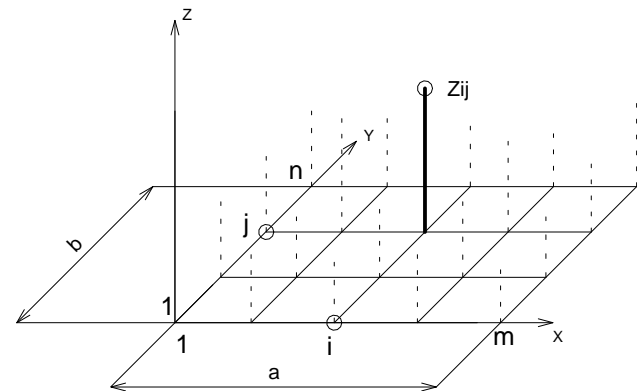
`j3` (4): боковые поверхности присутствуют.

`j5` (16): ребра основания и боковых поверхностей присутствуют.

`j6` (32): ребра верхней поверхности присутствуют.

`j7` (64): ребра верхней поверхности присутствуют, верхняя поверхность не является сглаженной,

`j8` (128): все ребра остроконечные, однако поверхность сглаженная.



*Ограничения на параметры:*

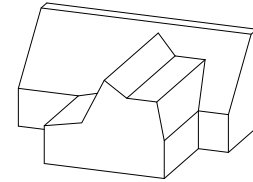
$n \geq 3$ ,  $m \geq 0$

*Пример:*

```

MASS  "Побелка", "Побелка", "Побелка",
      15, 12, 117, -5.0,
      0, 12, 0, 15,
      8, 12, 0, 15,
      8, 0, 0, 15,
      13, 0, 0, 13,
      16, 0, 0, 13,
      19, 0, 0, 13,
      23, 0, 0, 13,
      24, 0, 0, 15,
      24, 12, 0, 15,
      28, 12, 0, 15,
      28, 20, 8, 13,
      28, 22, 8, 15,
      0, 22, 8, 15,
      0, 20, 8, 13,
      0, 12, 0, -1,

      0, 22, 8, 0,
      28, 22, 8, -1,
      23, 17, 5, 0,
      23, 0, 5, -1,
      13, 13, 1, 0,
      13, 0, 1, -1,
      16, 0, 7, 0,
      16, 19, 7, -1,
      0, 20, 8, 0,
      28, 20, 8, -1,
      19, 17, 5, 0,
      19, 0, 5, -1
    
```



## ЭЛЕМЕНТЫ ВИЗУАЛИЗАЦИИ

### LIGHT

```
LIGHT red, green, blue, shadow,
      radius, alpha, beta, angle_falloff,
      distance1, distance2,
      distance_falloff [[,] ADDITIONAL_DATA name1 = value1,
      name2 = value2, ...]
```

Источник света, расположенный в начале локальных координат, излучает разноцветный [красный, зеленый, синий] световой поток вдоль оси x. Свет распространяется параллельно оси x из точечного или дискового источника. Он имеет максимальную интенсивность в пределах угла alpha усеченного конуса, которая спадает до нуля в пределах угла beta. Угловой спад освещенности управляется параметром `angle_falloff`. (Нулевое значение параметра свидетельствует о существовании четкого контура, а положительное число - о более плавном переходе.) Действие источника света вдоль оси x управляется параметрами `distance1` и `distance2`. Параметр `distance_falloff` управляет спадом интенсивности с увеличением расстояния. (Нулевое значение свидетельствует о постоянной интенсивности, а чем больше положительное число, тем резче спад интенсивности.)

Преобразования координат влияют только на место расположения источника света и направленность освещения.

Параметр `shadow` влияет на отбрасывание тени.

0: источник света не отбрасывает тень.

1: источник света отбрасывает тень.

*Ограничения на параметры:*

$\alpha \leq \beta \leq 80^\circ$

Следующие комбинации параметров имеют специальное значение:

$\text{radius} = 0, \alpha = 0, \beta = 0$

Точечный источник света излучает свет во всех направлениях и не отбрасывает тень. Параметры `shadow` и `angle_falloff` игнорируются, предполагаются значения `shadow = 0, angle_falloff = 0`.

$\text{radius} > 0, \alpha = 0, \beta = 0$

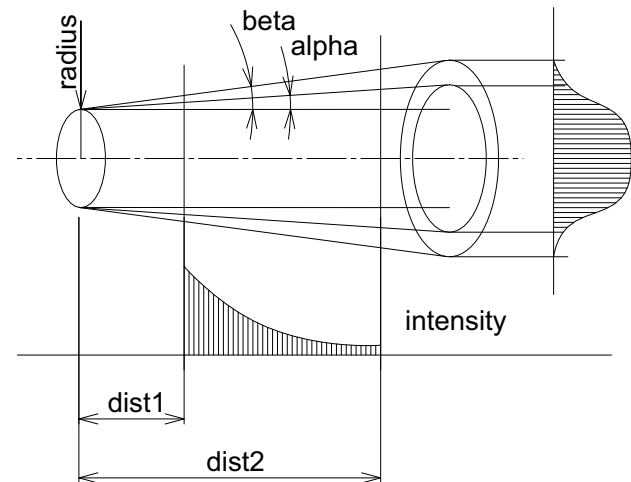
Направленный пучок света.

Определение источника света может содержать определение дополнительных данных после ключевого слова `ADDITIONAL_DATA`.

Дополнительные данные имеют имя (`namei`) и значение (`valuei`), которое может быть выражением любого типа, даже массив. Если имя строкового параметра заканчивается подстрокой "\_file", то его значение рассматривается как имя файла и он будет включен в архив проекта.

Для дополнительных данных может быть определен и использован различный смысл в ArchiCAD или расширениях ArchiCAD.

См. смысл параметров расширения *LightWorks* в <http://www.graphisoft.com/support/developer/documentation/LibraryDevDoc/10>.

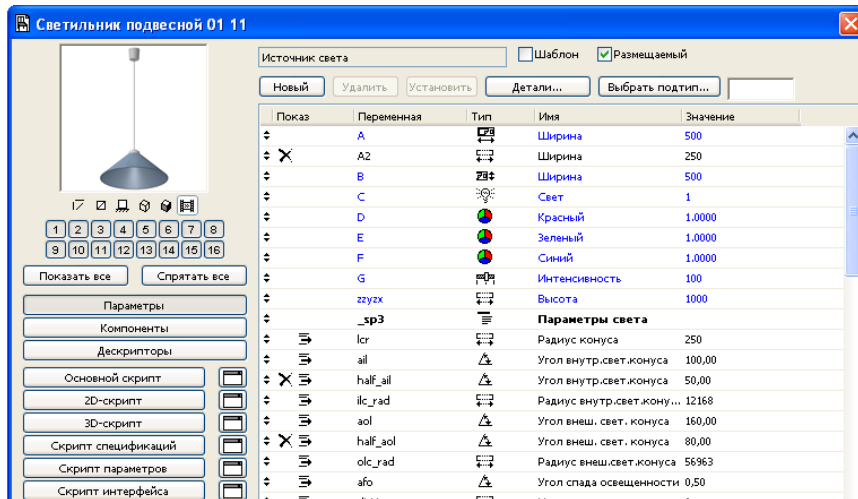


Пример:

```

LIGHT 1.0,0.2,0.3, ! RGB
      1,           ! тень отбрасывается
      1.0,        ! radius
      45.0,60.0,  ! angle1, angle2
      0.3,        ! angle_falloff
      1.0,10.0,  ! distance1, distance2
      0.2         ! distance_falloff
    
```

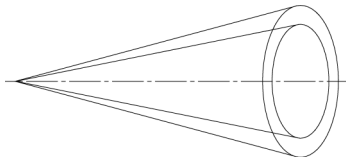
Диалоговое окно ArchiCAD ArchiFM установки параметров источников света:



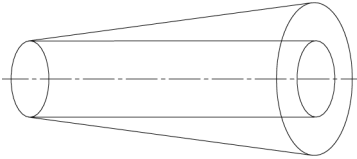
Частью соответствующего GDL-скрипта является:

```

IF C = 0 GOTO 10
LIGHT G/100*D, G/100*E, G/100*F, ! RGB
...
10:
    
```



$r = 0$ ,  $\alpha > 0$ ,  $\beta > 0$



$r > 0, \alpha = 0, \beta > 0$



$r > 0, \alpha = 0, \beta = 0$

Типы освещения в зависимости от параметров  $\alpha, \beta, \text{radius}$

## PICTURE

**PICTURE** expression, a, b, mask

Элемент типа рисунок, использующийся при фотосъемке.

Значение вычисленного выражения expression строкового типа обозначает имя файла, а значение числового типа указывает индекс рисунка, хранящегося в библиотечном элементе. Значение 0 индекса определяет рисунок образца библиотечного элемента. Другие рисунки могут быть запомнены в библиотечных элементах только при сохранении всего проекта или выбранных элементов, содержащих рисунки, в виде объекта ArchiCAD.

Ссылка на индексируемый рисунок не может использоваться в скрипте MASTER\_GDL, когда реквизиты объединяются в текущий набор реквизитов. Изображение вставляется в рамку, которая в любом из методов создания 3D-проекции трактуется как RECT.

mask = alpha + distortion. alpha: управление alpha-каналом.

0: alpha-канал не используется, рисунок является рамкой.

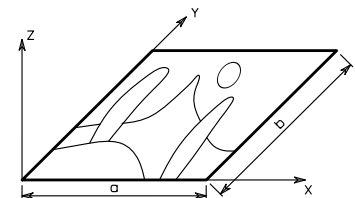
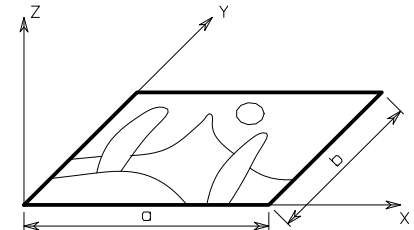
1: используется alpha-канал, части рисунка могут быть прозрачными.

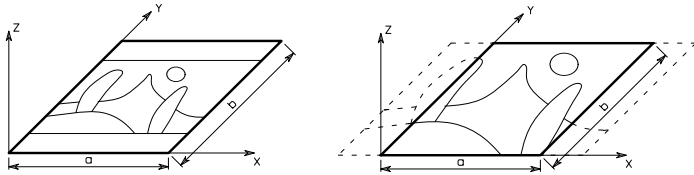
distortion: управление размещением рисунка.

0: поместить рисунок так, чтобы он занял всю рамку.

2: поместить весь рисунок по центру рамки так, чтобы сохранились его пропорции (часть рамки может оказаться пустой).

4: поместить рисунок по центру рамки так, чтобы вся рамка была заполнена и пропорции рисунка сохранились (часть рисунка может выйти за пределы рамки).





## ОБЪЕМНЫЕ ТЕКСТОВЫЕ ЭЛЕМЕНТЫ

### TEXT

**TEXT** d, 0, expression

Объемное представление значения выражения числового или строкового типа в текущем стиле.

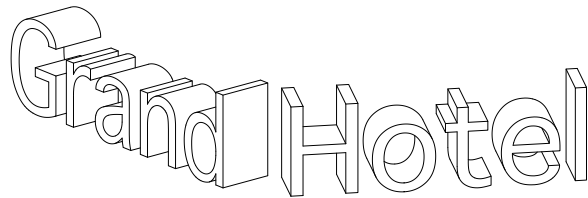
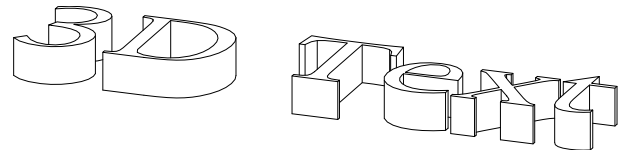
См. "[SET] STYLE" на стр. 157 и "DEFINE STYLE" на стр. 176.

d: толщина символа, измеряется в метрах.

В данной версии GDL второй параметр всегда равен 0.

*Примеры.*

```
DEFINE STYLE "aa" "New York", 3, 7, 0
SET STYLE "aa"
TEXT 0.005, 0, "3D-Text"
```



```
name = "Grand"
ROTX 90
ROTY -30
TEXT 0.003, 0, name
ADDX STW (name)/1000
ROTY 60
TEXT 0.003, 0, "Hotel"
```

**Примечание:** Для совместимости с 2D-скриптами GDL высота символов в предложении DEFINE STYLE всегда выражается в мм.

## RICHTEXT

**RICHTEXT** *x*, *y*,  
*height*, 0, *textblock\_name*

Объемное представление ранее определенного TEXTBLOCK. Для получения дополнительной информации см. [“Определение текстового блока” на стр. 178](#).

*x*, *y*: координаты X-Y расположения обогащенного текста (richtext).

*height*: толщина символов в метрах.

*textblock\_name*: имя ранее определенного TEXTBLOCK.

В текущей версии GDL четвертый параметр всегда равен нулю.

## ПРИМИТИВНЫЕ ЭЛЕМЕНТЫ

Примитивами трехмерной структуры данных являются VERT, VECT, EDGE, PGON и BODY. Тела представляются поверхностями и их пересечениями. Информация для получения 3D-разрезов поступает из пересечения поверхностей.

Индексирование элементов начинается с 1. Любое из предложений BASE или BODY (неявное подразумевающее предложение BASE) приводит к установке индекса в 1. Для каждого ребра запоминаются индексы прилегающих многоугольников (не более 2). Ориентации ребер определяется последовательностью задания его двух вершин.

Многоугольники представляются в виде списка ребер с их ориентацией, включая и их индексы. Эти числа могут иметь отрицательный префикс. Это означает, что ребро используется в противоположном направлении. В многоугольниках могут быть отверстия. Нулевое значение индекса в списке ребер свидетельствует о начале нового отверстия. В отверстиях не может быть других отверстий. Одно ребро может принадлежать 0-2 многоугольникам. В случае замкнутых тел ориентация многоугольника считается правильной, если ребро имеет различные префиксы в списке ребер двух многоугольников.

Векторы нормали многоугольников запоминаются отдельно. В случае замкнутых тел они направлены изнутри тела наружу. Ориентация списка ребер направлена против часовой стрелки (математически положительная), если Вы смотрите на них извне. Ориентация отверстий противоположна их многоугольникам. Векторы нормали открытого тела должны быть направлены в одну и ту же его сторону.

Для определения внутренней и наружной сторон тела оно должно быть замкнутым. Простейшим определением замкнутого тела является следующее: любое ребро имеет ровно два прилегающих многоугольника.

Для открытых тел алгоритмы определения разрезов, удаления невидимых линий и построения реалистических изображений менее неэффективны. Любой составной трехмерный элемент с правильными параметрами является замкнутым телом с точки зрения его внутренней трехмерной структуры данных.

Нахождение линии контура основывается на битах статуса ребер и прилегающих многоугольников. Для составных криволинейных элементов они устанавливаются автоматически, однако в случае определения примитивных элементов ответственность за правильную установку таких битов лежит на Вас.

В случае упрощенного определения (в PGON vect = 0 или статус < 0) примитивы, на которые имеются ссылки, должны стоять перед теми, которые на них ссылаются. В данном случае рекомендуется следующий порядок определения примитивов:

VERT (TEVE)  
 EDGE  
 (VECT)  
 PGON (PIPG)  
 COOR  
 BODY

Поиск прилегающих к ребру многоугольников производится при выполнении предложения

Нумерация элементов VERT, EDGE, VECT и PGON производится относительно последнего явно или неявно заданного предложения BASE.

Значения статуса используются для запоминания специальной информации о примитивах. Обычно каждый отдельный бит статуса имеет независимое значение, однако имеются исключения.

Приводимые значения статусов могут складываться. Комбинации битов статуса, отличные от приводимых, зарезервированы для внутреннего использования. По умолчанию любой статус принимается равным 0.

## VERT

**VERT**  $x, y, z$

Вершина в плоскости  $x$ - $y$ - $z$ , определяемая тремя координатами.

## TEVE

**TEVE**  $x, y, z, u, v$

Расширение предложения VERT, включающее определение координат текстуры. Может использоваться вместо предложения VERT в том случае, когда следует указать определенные координаты начала текстуры вместо автоматического нанесения текстуры, выполняемого системой ArchiCAD (см. предложение "*COOR*" на стр. 98).

$x, y, z$ : координаты вершины.

$u, v$ : координаты текстуры для этой вершины.

Для каждой вершины тела должны быть определены координаты  $(u,v)$ , и каждая вершина может иметь только одну такую пару координат текстуры. Если в определении тела используется оба предложения и VERT и TEVE, то введение координат  $(u,v)$  в таком случае уже не действуют.

**Примечание:** Использование координат  $(u,v)$  действительно только для построения фотореалистических изображений, но отнюдь не для нанесения векторной штриховки.

## VECT

**VECT**  $x, y, z$

Вектор нормали многоугольника, определяемый тремя координатами. В случае упрощенного определения ( $vect=0$  в PGON), можно не использовать это предложение.



## EDGE

**EDGE** `vert1, vert2, pgon1, pgon2, status`

Определение ребра.

`vert1, vert2`: индексы двух крайних точек. Индексы `vert1` и `vert2` должны отличаться и ссылаться на ранее определенные `VERT`.

`pgon1, pgon2`: индексы прилегающих многоугольников. Ноль и отрицательное число имеют следующие значения:

0: самое крайнее или отдельно стоящее ребро.

< 0: будет произведен поиск возможных прилегающих многоугольников.

*Биты статуса:*

1: невидимое ребро.

2: ребро криволинейной поверхности.

*Зарезервированные значения битов статуса для будущего использования:*

4: первое ребро криволинейной поверхности (только вместе с 2).

8: последнее ребро криволинейной поверхности (только вместе с 2).

16: ребро является дугой.

32: первый отрезок дуги (только вместе с 16).

64: последний отрезок дуги (только вместе с 16).

## PGON

**PGON** `n, vect, status, edge1, edge2, ... edgen`

Определение многоугольника.

`n`: количество ребер в списке.

`vect`: индекс вектора нормали. Должен ссылаться на ранее определенный `VECT`.

**Примечание:** Если `vect = 0`, то ArchiCAD вычислит вектор нормали при анализе.

Индексы `edge1, edge2, ... edgen` должны ссылаться на ранее определенные `EDGE`. Индекс 0 обозначает начало или конец определения отверстия.

Отрицательный индекс изменяет в многоугольнике направление запомненного вектора нормали или ребра на противоположное. (Сам вектор или ребро не изменяет направления; другие многоугольники могут ссылаться на них с использованием их исходной ориентации, задавая положительный индекс).

*Биты статуса:*

- 1: невидимый многоугольник.
- 2: многоугольник криволинейной поверхности.
- 16: вогнутый многоугольник.
- 32: многоугольник с отверстием(ями).
- 64: выпуклое отверстие(я) (только вместе с 32).

*Зарезервированные значения битов статуса для будущего использования:*

- 4: первый многоугольник криволинейной поверхности (только вместе с 2).
- 8: последний многоугольник криволинейной поверхности (только вместе с 2).

Если значение статуса отрицательное, то ArchiCAD сам вычислит статус многоугольника (вогнутого типа или многоугольника с отверстием).

$n = 0$  допускается в особых случаях.

## PIPG

**PIPG** expression, a, b, mask, n, vect,  
status,  
edge1, edge2, ... edgen

Определение многоугольника рисунка. Первые 4 параметра аналогичны элементу PICTURE, а остальные аналогичны элементу PGON.

## COOR

**COOR** wrap, vert1, vert2, vert3, vert4

Локальная система координат для BODY для нанесения штриховки и текстуры.

wgap: форма поверхности и типы проекций.

*Формы поверхности:*

- 1: плоскость,
- 2: куб,
- 3: цилиндр,
- 4: сфера,

5: аналогична цилиндрической поверхности, однако при визуализации нижняя и верхняя проекции предполагаются сферическими.

Типы проекций:

256: штриховка всегда начинается в начале локальной системы координат.

1024: квадратичная проекция текстуры (рекомендуется).

2048: линейная проекция текстуры, за основу при проецировании берется среднее расстояние.

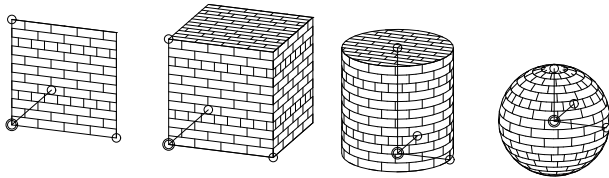
4096: линейная проекция текстуры на основе нормальной триангуляции.

**Примечание:** Последние три значения действуют только при непосредственном определении координат текстуры (см. "TEVE" на стр. 96).

vert1: индекс вершины VERT, представляющей начало локальной системы координат.

vert2, vert3, vert4: индексы вершин VERT определяющих 3 оси координат.

Если индексы VERT используются только для определения локальной системы координат, то укажите их со знаком минус.

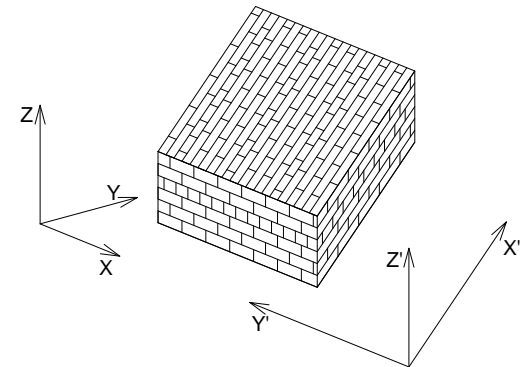


Пример задания осей координат текстуры:

CSLAB\_ "Кирпич облицовочный", "Кирпич облицовочный", "Кирпич облицовочный",

```
4, 0.5,
0, 0, 0, 15,
1, 0, 0, 15,
1, 1, 1, 15,
0, 1, 1, 15
```

```
BASE
VERT 1, 0, 0 !#1
VERT 1, 1, 1 !#2
VERT 0, 0, 0 !#3
VERT 1, 0, 1 !#4
COOR 2, -1, -2, -3, -4
BODY 1
```



## BODY

**BODY** status

Формирует тело из определенных ранее примитивов.

*Биты статуса:*

- 1: замкнутое тело.
- 2: тело включает криволинейную поверхность(и).
- 4: планарная модель: когда делается разрез тела, то в плоскости сечения отсутствует поверхность.
- 32: тело всегда отбрасывает тень независимо от автоматически выбираемого алгоритма.
- 64: тело никогда не отбрасывает тень.

Если не определено ни 32, ни 64, то используется алгоритм автоматического определения отбрасывания теней.

См. *“SHADOW”* на стр. 160.

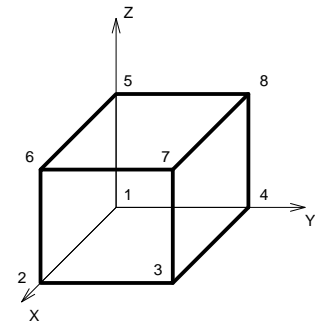
Если значение статуса отрицательное, то ArchiCAD сам вычислит статус тела.

*Пример:*

1: Полное описание

```

VERT 0.0, 0.0, 0.0      !#1
VERT 1.0, 0.0, 0.0      !#2
VERT 1.0, 1.0, 0.0      !#3
VERT 0.0, 1.0, 0.0      !#4
VERT 0.0, 0.0, 1.0      !#5
VERT 1.0, 0.0, 1.0      !#6
VERT 1.0, 1.0, 1.0      !#7
VERT 0.0, 1.0, 1.0      !#8
EDGE 1, 2, 1, 3, 0      !#1
EDGE 2, 3, 1, 4, 0      !#2
EDGE 3, 4, 1, 5, 0      !#3
EDGE 4, 1, 1, 6, 0      !#4
EDGE 5, 6, 2, 3, 0      !#5
EDGE 6, 7, 2, 4, 0      !#6
EDGE 7, 8, 2, 5, 0      !#7
EDGE 8, 5, 2, 6, 0      !#8
EDGE 1, 5, 6, 3, 0      !#9
EDGE 2, 6, 3, 4, 0      !#10
EDGE 3, 7, 4, 5, 0      !#11
EDGE 4, 8, 5, 6, 0      !#12
VECT 1.0, 0.0, 0.0      !#1
VECT 0.0, 1.0, 0.0      !#2
    
```



```

VECT 0.0, 0.0, 1.0      !#3
PGON 4, -3, 0, -1, -4, -3, -2  !#1      !VERT1,2,3,4
PGON 4, 3, 0, 5, 6, 7, 8      !#2      !VERT5,6,7,8
PGON 4, -2, 0, 1, 10, -5, -9   !#3      !VERT1,2,5,6
PGON 4, 1, 0, 2, 11, -6, -10  !#4      !VERT2,3,6,7
PGON 4, 2, 0, 3, 12, -7, -11  !#5      !VERT3,4,7,8
PGON 4, -1, 0, 4, 9, -8, -12   !#6      !VERT1,4,5,8
BODY 1                        ! KUB

```

2: (нет прямых ссылок на многоугольники или векторы; ArchiCAD их вычислит сам)

```

VERT 0.0, 0.0, 0.0      !#1
VERT 1.0, 0.0, 0.0      !#2
VERT 1.0, 1.0, 0.0      !#3
VERT 0.0, 1.0, 0.0      !#4
VERT 0.0, 0.0, 1.0      !#5
VERT 1.0, 0.0, 1.0      !#6
VERT 1.0, 1.0, 1.0      !#7
VERT 0.0, 1.0, 1.0      !#8
EDGE 1, 2, -1, -1, 0     !#1
EDGE 2, 3, -1, -1, 0     !#2
EDGE 3, 4, -1, -1, 0     !#3
EDGE 4, 1, -1, -1, 0     !#4
EDGE 5, 6, -1, -1, 0     !#5
EDGE 6, 7, -1, -1, 0     !#6
EDGE 7, 8, -1, -1, 0     !#7
EDGE 8, 5, -1, -1, 0     !#8
EDGE 1, 5, -1, -1, 0     !#9
EDGE 2, 6, -1, -1, 0     !#10
EDGE 3, 7, -1, -1, 0     !#11
EDGE 4, 8, -1, -1, 0     !#12
PGON 4, 0, -1, -1, -4, -3, -2  !#1
                                !VERT1,2,3,4
PGON 4, 0, -1, 5, 6, 7, 8      !#2
                                !VERT5,6,7,8
PGON 4, 0, -1, 1, 10, -5, -9   !#3
                                !VERT1,2,5,6
PGON 4, 0, -1, 2, 11, -6, -10  !#4
                                !VERT2,3,6,7
PGON 4, 0, -1, 3, 12, -7, -11  !#5
                                !VERT3,4,7,8
PGON 4, 0, -1, 4, 9, -8, -12   !#6
                                !VERT1,4,5,8
BODY -1                        !KUB

```

## BASE

### BASE

Сбрасывает счетчики примитивных элементов (VERT, TEVE, VECT, EDGE, PGON и PIPG). Неявно используется после каждого определения составного элемента.

## ПЛОСКОСТИ СЕЧЕНИЯ В 3D

### CUTPLANE

```
CUTPLANE [x, y, z [, side [, status]]]
    [statement1 ... statementn]
CUTEND
```

*или*

```
CUTPLANE{2} angle [, status]
    [statement1 ... statementn]
CUTEND
```

Строит плоскости сечения и удаляет отсеченные элементы. CUTPLANE может иметь различное число параметров.

*Если CUTPLANE имеет следующее количество параметров:*

0: плоскость x-y;

1: плоскость сечения пересекает ось x, angle - угол между плоскостью сечения и плоскостью x-y;

2: плоскость сечения параллельна оси z и пересекает оси x и y в заданных точках;

3: пересекает оси x, y и z в заданных точках;

4: первые три параметра определяются также как и выше, с дополнительным следующим значением параметра side:

side = 0: удаляет все фигуры, расположенные выше плоскости (по умолчанию);

side = 1: удаляет фигуры, расположенные ниже плоскости сечения; в случае x-y, x-z, y-z, в случаях, когда плоскость сечения совпадает с плоскостями x-y, y-z или x-z, удаляются элементы, расположенные в отрицательном направлении по третьей оси координат.

При сечении (когда не задан параметр side) удаляются элементы фигуры, расположенные выше плоскости сечения. Если первые три параметра определяют плоскость x-y, x-z или y-z (например, 1.0, 1.0, 0.0 определяет плоскость x-y), то удаляются элементы фигуры, расположенные в положительном направлении третьей оси координат.

Между командами CUTPLANE и CUTEND может быть расположено любое количество команд любого типа. CUTPLANE также может содержаться в макросе.

Параметры команды CUTPLANE принимают значения в текущей системе координат.

Преобразования координат, заданные между CUTPLANE и CUTEND, не действуют на определенную этими операторами плоскость сечения, однако любая из последующих плоскостей сечения будет строиться с учетом текущего преобразования. Таким образом, для определения CUTPLANE можно воспользоваться любым количеством преобразований, однако перед последующим определением подлежащих рассечению фигур удалите эти преобразования.

Пары команд CUTPLANE–CUTEND могут вкладываться друг в друга, даже внутри циклов. Если отсутствует CUTEND, то соответствующий CUTPLANE будет действовать на все фигуры до конца скрипта.

Команды CUTPLANE внутри макроса действуют только на фигуры этого макроса, даже если отсутствует CUTEND.

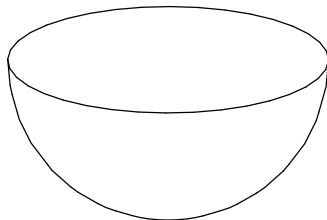
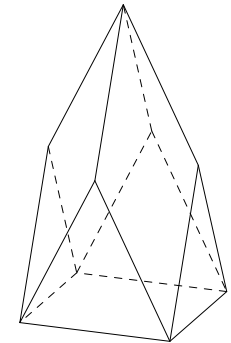
Если между командами CUTPLANE и CUTEND вызывается макрос, то все фигуры, построенные при выполнении этого макроса, будут рассечены.

На поверхности сечения распространяются текущие установки покрытия, цвета пера и образца штриховки.

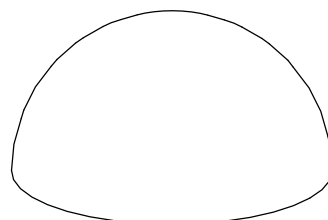
**Примечание:** Если CUTPLANE не закрыт командой CUTEND, в наихудшем случае будут удалены полностью все построенные фигуры. Вот почему всегда будет выдаваться сообщение-предупреждение об отсутствии команды CUTEND. Если преобразования, используемые только для определения расположения плоскости сечения, не удалены сразу после ее определения, то получив неправильный результат, Вы можете подумать, что неправильно расположена плоскость сечения, когда, на самом деле, произошло перемещение фигур.

*Примеры.*

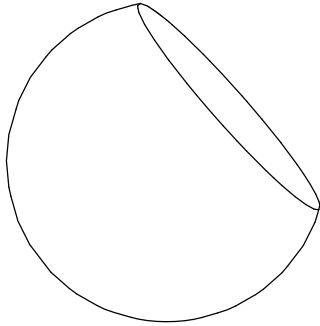
```
CUTPLANE 2, 2, 4
CUTPLANE -2, 2, 4
CUTPLANE -2, -2, 4
CUTPLANE 2, -2, 4
  ADD -1, -1, 0
  BRICK 2, 2, 4
  DEL 1
CUTEND
CUTEND
CUTEND
CUTEND
```



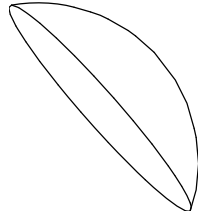
```
CUTPLANE
  SPHERE 2
CUTEND
```



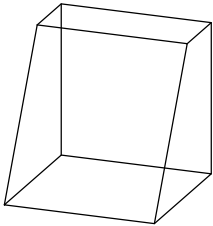
```
CUTPLANE 1, 1, 0, 1
  SPHERE 2
CUTEND
```



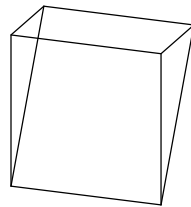
```
CUTPLANE 1.8, 1.8,
          1.8
SPHERE 2
CUTEND
```



```
CUTPLANE 1.8, 1.8,
          1.8, 1
SPHERE 2
CUTEND
```



```
CUTPLANE 60
BRICK 2, 2, 2
CUTEND
```



```
CUTPLANE -120
BRICK 2, 2, 2
CUTEND
```

## CUTPOLY

```
CUTPOLY n,
          x1, y1, ... xn, yn
          [, x, y, z]
          [statement1
          statement2
          ...
          statementn]
CUTEND
```



Как и в предложении CUTPLANE параметры задаются в текущей системе координат. Многоугольник сечения не должен содержать самопересечений. Сечение производится в направлении оси z или согласно необязательному вектору (x,y,z).

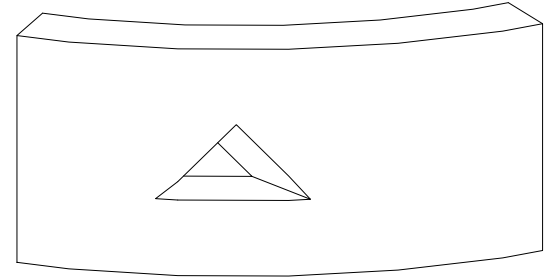
Параметры определяют бесконечную многоугольную призму. Направление сечения совпадает с направлением призмы. Все элементы фигуры, попавшие внутрь призмы, удаляются.

*Примеры.*

```

ROTX 90
MULZ -1
CUTPOLY 3,
    0.5, 1,
    2, 2,
    3.5, 1,
    -1.8, 0, 1
DEL 1
BPRISM "Кирпич красный", "Кирпич красный", "Кирпич красный",
    4, 0.9, 7,
    0.0, 0.0, 15,
    6.0, 0.0, 15,
    6.0, 3.0, 15,
    0.0, 3.0, 15
CUTEND

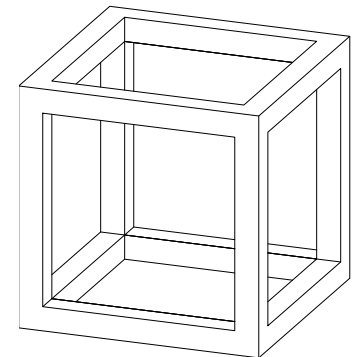
```



```

A=1.0
D=0.1
CUTPOLY 4,
    D, D,
    A-D, D,
    A-D, A-D,
    D, A-D
ROTX 90
CUTPOLY 4,
    D, D,
    A-D, D,
    A-D, A-D,
    D, A-D
DEL 1
ROTY 90
CUTPOLY 4,
    D, D,
    A-D, D,
    A-D, A-D,
    D, A-D

```

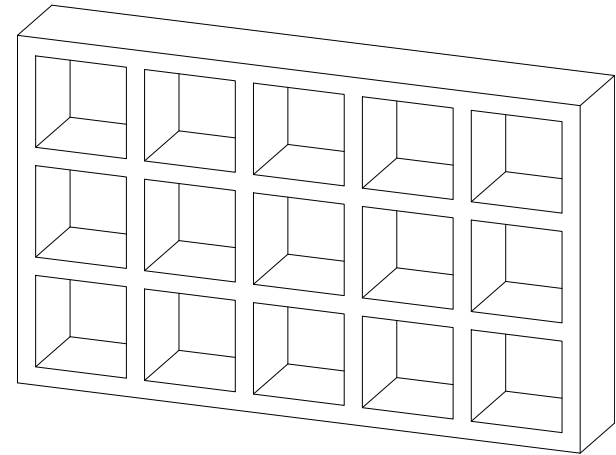


```

DEL 1
BLOCK A, A, A
CUTEND
CUTEND
CUTEND

ROTX 90
FOR I=1 TO 3
  FOR J=1 TO 5
    CUTPOLY 4,
      0, 0, 1, 0,
      1, 1, 0, 1
    ADDX 1.2
  NEXT J
  DEL 5
  ADDY 1.2
NEXT I
DEL NTR()-1
ADD -0.2, -0.2, 0
BRICK 6.2, 3.8, 1
FOR K=1 TO 15
  CUTEND
NEXT K
DEL TOP

```



## CUTPOLYA

```

CUTPOLYA n, status, d,
  x1, y1, mask1, ... xn, yn, maskn [,
  x, y, z]
  [statement1
  statement2
  ...
  statementn]

```

### CUTEND

Аналогично предложению CUTPOLY, однако предоставляет возможность управления видимостью ребер создаваемых поверхностей. Сечение имеет форму многоугольной полубесконечной призмы. Если конец фигуры сечения входит внутрь тела, то она отсекает соответствующую область.

status: трактует создаваемые многоугольники сечения.

1: для создания граней и ребер многоугольника при рассечении используются реквизиты самого тела.

2: создаваемые многоугольники сечения трактуются как обычные многоугольники.

d: расстояние от конца полубесконечной призмы и до начала локальной системы координат.

d = 0 определяет сечение бесконечной призмой.

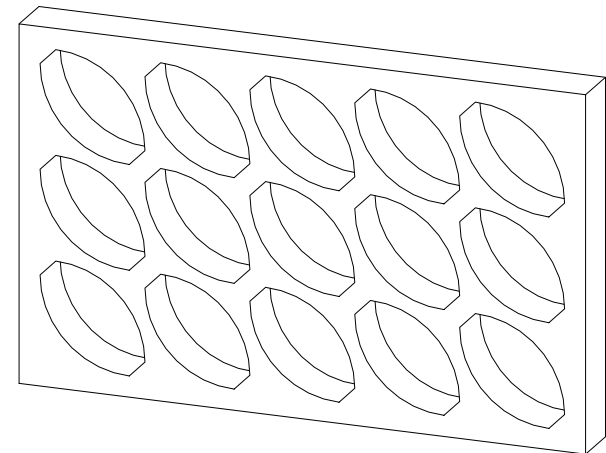
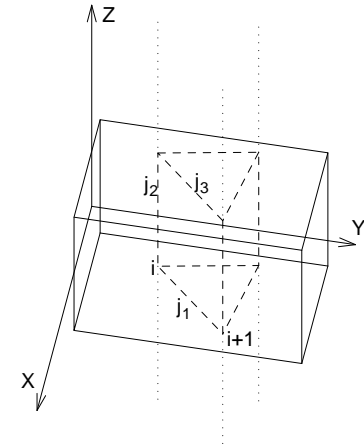
maski: как и в предложении PRISM\_.

maski = j1 + 2 \* j2 + 4 \* j3 + 64 \* j7

Пример:

```

ROTX 90
FOR I=1 TO 3
  FOR J=1 TO 5
    CUTPOLYA 6, 1, 0,
      1, 0.15, 5,
      0.15, 0.15, 90,
      0, 90, 4007,
      0, 0.85, 5,
      0.85, 0.85, 90,
      0, 90, 4007
    ADDX 1
  NEXT J
  DEL 5
  ADDY 1
NEXT I
DEL NTR() -1
ADD -0.2, -0.2, 0
BRICK 5.4, 3.4, 0.5
FOR K=1 TO 15
  CUTEND
NEXT K
DEL TOP
    
```



## CUTSHAPE

```
CUTSHAPE d [, status]
[statement1 statement2 ... statementn]
```

CUTEND

status: трактует создаваемые многоугольники сечения. Если параметр не задан (в целях совместимости), то по умолчанию он принимает значение 3.

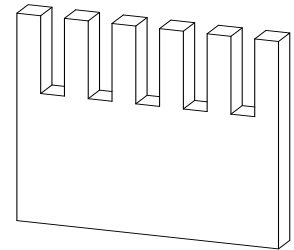
status = j1 + 2\*j2

j1: для создания граней и ребер многоугольника при рассечении используются реквизиты самого тела.

j2: создаваемые многоугольники сечения тракуются как обычные многоугольники.

*Пример:*

```
FOR I = 1 TO 5
  ADDX 0.4 * I
  ADDZ 2.5
  CUTSHAPE 0.4
  DEL 2
  ADDX 0.4
NEXT I
DEL TOP
BRICK 4.4, 0.5, 4
FOR I = 1 TO 5
  CUTEND
NEXT I
```



## CUTFORM

```
CUTFORM n, method, status,
  rx, ry, rz, d,
  x1, y1, mask1,
  ...
  xn, yn, maskn
```

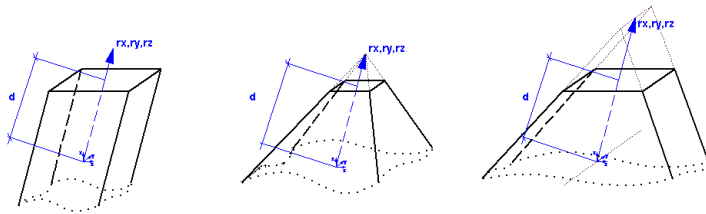
Аналогично определению предложения CUTPOLYA, однако предоставляет дополнительную возможность управления формой и протяженностью тела сечения.

method: управляет формой тела сечения.

1: имеет форму призмы.

2: имеет форму пирамиды.

3: клинообразное тело сечения. Верхнее ребро клина направлено параллельно оси Y и оно расположено в gx, gy, gz (gy игнорируется.)



status: управляет протяженностью тела сечения и трактовкой создаваемых многоугольников сечения и новых ребер.

status = j1 + 2\*j2 + 8\*j4 + 16\*j5 + 32\*j6 + 64\*j7 + 128\*j8

j1: для создания граней и ребер многоугольника при рассечении используются реквизиты самого тела.

j2: создаваемые многоугольники сечения трактуются как обычные многоугольники.

j4, j5: определяет ограничения на сечение:

j4 = 0 и j5 = 0: конечное сечение,

j4 = 0 и j5 = 1: полубесконечное сечение,

j4 = 1 и j5 = 1: бесконечное сечение.

j6: в качестве результата выбирается то, что получается при пересечении исходного тела с телом сечения, а не их разность. (Может использоваться только с командой CUTFORM.)

j7: ребра, создаваемые нижней частью тела сечения, будут невидимыми.

j8: ребра, создаваемые верхней частью тела сечения, будут невидимыми.

gx, gy, gz: определяет направление сечения, если фигура сечения имеет форму призмы, или вершину пирамиды, если тело сечения пирамидальное.

d: определяет расстояние вдоль gx, gy, gz до конца сечения. Если сечение бесконечное, то этот параметр не действует. Если сечение конечное, то тело сечения начинается в начале локальной системы координат и заканчивается на расстоянии d вдоль направления, определенного gx, gy, gz.

Если сечение полубесконечное, то тело сечения начинается в точке, расположенной на расстоянии d вдоль направления, определенного by gx, gy, gz, а направление полубесконечного сечения является противоположным направлению, определяемому gx, gy, gz.

mask: определяет видимость ребер тела сечения.

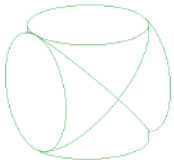
maski = j1 + 2\*j2 + 4\*j3 + 8\*j4 + 16\*j5 + 64\*j7

- j1: многоугольник будет создавать видимое ребро при входе в высекаемое тело.
- j2: продольное ребро фигуры сечения будет видимым.
- j3: многоугольник будет создавать видимое ребро при выходе их высекаемого тела.
- j4: нижнее ребро фигуры сечения будет видимым.
- j5: верхнее ребро фигуры сечения будет видимым.
- j7: управляет видимостью продольного ребра, которая зависит от точки наблюдения.

## КОМАНДЫ НАД ОБЪЕМНЫМИ ЭЛЕМЕНТАМИ

GDL обладает возможностью выполнять специальные 3D-операции над объемными элементами, представленными в виде групп. Далее приводятся эти операции:

- **ADDGROUP**: производится объединение двух объемных элементов.



- **SUBGROUP**: производится вычитание одного объемного элемента из другого.



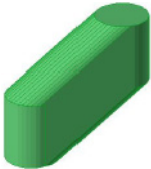
- **ISECTGROUP**: выбирается общая часть двух объемных элементов.



- **ISECTLINES**: производится вычисление линий пересечения двух объемных элементов.



- **SWEEPGROUP**: поворот объемного элемента вдоль тела.



Объемный элемент GDL состоит из одного или более "кусков", которые представляются в модели в виде отдельных тел. Кусок имеет точно одну внешнюю оболочку и может содержать полости. (Полости могут быть описаны как "отрицательные" оболочки внутри куска.) Тело на рисунке ниже состоит из двух кусков, причем один из них содержит полость.

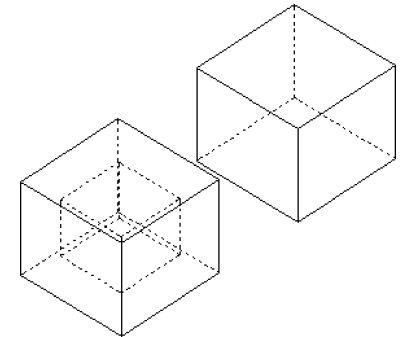
Такие тела GDL, как BLOCK, SPHERE и т.д. представляются как внешние оболочки в группах. С помощью приводимой ниже конструкции пользователь может поместить в тело более одной оболочки (обратите внимание на предложение BODY -1):

```
GROUP "myGroup"
  BLOCK 1,1,1
  BODY -1
  ADDX 1
  BLOCK 1,1,1
ENDGROUP
```

Описанное выше тело содержит два куска, каждый из которых состоит из одной оболочки. Пустоты могут быть определены с помощью примитивов, либо могут появиться в результате выполнения операции вычитания одного объемного элемента из другого (например, вычитание из куба другого куба, который размещается внутри первого).

См. также "*Примитивные элементы*" на стр. 95.

Хотя групповые операции предназначены, прежде всего, для работы с объемными элементами, однако они также могут быть применены к моделям, представленным в виде поверхностей или каркасов, а также к гибридным моделям. (Гибридные модели - это в основном поверхности, которые могут содержать ребра без соседних граней.) Результаты применения этих операций представлены в приводимых далее таблицах:



**Объединение (база » оператор)**

	<b>объемная база</b>	<b>база-поверхность</b>	<b>каркасная база</b>	<b>гибридная база</b>
<b>объемный оператор</b>	объемный результат	результат-поверхность (слияние)	каркасный результат (слияние)	гибридный результат (слияние)
<b>оператор-поверхность</b>	результат-поверхность (слияние)	результат-поверхность (слияние)	гибридный результат (слияние)	гибридный результат (слияние)
<b>каркасный оператор</b>	каркасный результат (слияние)	гибридный результат (слияние)	каркасный результат (слияние)	гибридный результат (слияние)
<b>гибридный оператор</b>	гибридный результат (слияние)	гибридный результат (слияние)	гибридный результат (слияние)	гибридный результат (слияние)

**Разность(база\оператор)**

	<b>объемная база</b>	<b>база-поверхность</b>	<b>каркасная база</b>	<b>гибридная база</b>
<b>объемный оператор</b>	объемный результат	результат-поверхность	каркасный результат	гибридный результат
<b>оператор-поверхность</b>	объемная база (ничего не происходит)	база-поверхность (ничего не происходит)	каркасная база (ничего не происходит)	гибридная база (ничего не происходит)
<b>каркасный оператор</b>	объемная база (ничего не происходит)	база-поверхность (ничего не происходит)	каркасная база (ничего не происходит)	гибридная база (ничего не происходит)
<b>гибридный оператор</b>	объемная база (ничего не происходит)	база-поверхность (ничего не происходит)	каркасная база (ничего не происходит)	гибридная база (ничего не происходит)

**пересечение (база « оператор)**

	<b>объемная база</b>	<b>база-поверхность</b>	<b>каркасная база</b>	<b>гибридная база</b>
<b>объемный оператор</b>	объемный результат	результат-поверхность	каркасный результат	гибридный результат
<b>оператор-поверхность</b>	результат-поверхность	результат-пусто	результат-пусто	результат-пусто
<b>каркасный оператор</b>	каркасный результат	результат-пусто	результат-пусто	результат-пусто
<b>гибридный оператор</b>	гибридный результат	результат-пусто	результат-пусто	результат-пусто



**Пересечение линий (база « оператор »)**

	<b>объемная база</b>	<b>база-поверхность</b>	<b>каркасная база</b>	<b>гибридная база</b>
<b>объемный оператор</b>	каркасный результат	каркасный результат	результат-пусто	каркасный результат
<b>оператор-поверхность</b>	каркасный результат	результат-пусто	результат-пусто	результат-пусто
<b>каркасный оператор</b>	результат-пусто	результат-пусто	результат-пусто	результат-пусто
<b>гибридный оператор</b>	каркасный результат	результат-пусто	результат-пусто	результат-пусто

**Вытягивание**

<b>объемное тело</b>	<b>поверхность</b>	<b>каркасное тело</b>	<b>гибридное тело</b>
правильный результат	база-поверхность (ничего не происходит)	каркасная база (ничего не происходит)	гибридная база (ничего не происходит)

Поверхности могут создаваться непосредственно с помощью команды `MODEL SURFACE`, или опосредованно путем опускания из модели не являющиеся соседними многоугольники-грани. Каркасные модели создаются либо с использованием предложения `MODEL WIRE`, либо определением объектов без многоугольников-граней. Гибридные модели могут создаваться только опосредованно путем опускания из модели являющиеся соседними многоугольники-грани.

В подавляющем большинстве случаев требуемая модель является объемным телом. Тела GDL появляются в качестве оболочек в определениях групп. В связи с этим для достижения быстрого и надежного выполнения операций критически важным является получение правильных с геометрической точки зрения оболочек. Необходимость оперирования созданными объектами требует загрузки интерпретатора GDL, что приводит к увеличению времени на выполнение требуемой операции. Основное правило, относящееся к эффективному использованию операций группирования GDL, может быть кратко выражено следующим образом: *моделируйте таким образом, чтобы создаваемые Вами объекты действительно имели бы физические аналоги.* С практической точки зрения все это можно выразить в виде следующих рекомендаций:

- избегайте построения самопересекающихся объектов;
- избегайте построения самокасающихся объектов (используйте небольшие зазоры);
- избегайте построения объектов с нулевыми размерами (используйте небольшую толщину).

Согласно всего сказанного выше эти правила должны удовлетворяться при построении оболочек (определяемых посредством тел), а не при построении объемных элементов (определяемых посредством групп). (Объемный элемент, создаваемый скриптом с использованием представленной выше конструкции группы, смоделирован правильно, так как составляющие его оболочки касаются друг друга, но не самих себя, и сами они являются правильными с геометрической точки зрения.)

## GROUP

**GROUP** "name"

Начало определения группы. Все тела, определяемые до следующего предложения **ENDGROUP**, входят в состав группы под именем "name". Определение группы не подразумевает ее фактического создания (размещения); группа может использоваться в групповых операциях или может быть явно размещена с помощью команды **PLACEGROUP**. Определение группы нельзя вкладывать, однако можно включать макровыводы, содержащие определения групп и команды **PLACEGROUP**, использующие другие группы.

Имена групп должны быть уникальными в пределах текущего скрипта. Преобразования и секущие плоскости, за даваемые за пределами определения группы, не оказывают влияния на составные части группы; преобразования и секущие плоскости, используемые внутри группы, не оказывают влияние на тела за пределами определения группы. Определения групп являются прозрачными для реквизитов, которые **DEFINE** или **SET** (перья, покрытия, штриховка); реквизиты, определенные/установленные до определения группы, а также те, которые определены/установлены внутри группы, являются действительными.

## ENDGROUP

**ENDGROUP**

Завершение определения группы.

## ADDGROUP

**ADDGROUP** (g\_expr1, g\_expr2)

## SUBGROUP

**SUBGROUP** (g\_expr1, g\_expr2)

## ISECTGROUP

**ISECTGROUP** (g\_expr1, g\_expr2)

## ISECTLINES

**ISECTLINES** (g\_expr1, g\_expr2)

Операции над группами: объединение, вычитание, пересечение, пересечение линий. Возвращаемое значение - новая группа, которая может быть размещена по команде **PLACEGROUP**, запомнена в переменной или использована в качестве параметра в другой группе. Групповая операция может быть выполнена над ранее определенными группами или группами, получившимися в результате выполнения любой другой групповой операции. g\_expr1, g\_expr2 являются выражениями группового типа. Выражения группового типа - это либо имена групп (строковые выражения), либо переменные группового типа, либо любая их комбинация в операциях, которые в качестве своего результата возвращают группу.

## PLACEGROUP

**PLACEGROUP** *g\_expr*

Размещение группы - это именно та операция, в результате выполнения которой фактически создаются тела. При этом действительными являются плоскости сечений и преобразования координат; происходит вычисление группового выражения и результирующие тела запоминаются в структуре 3D-данных.

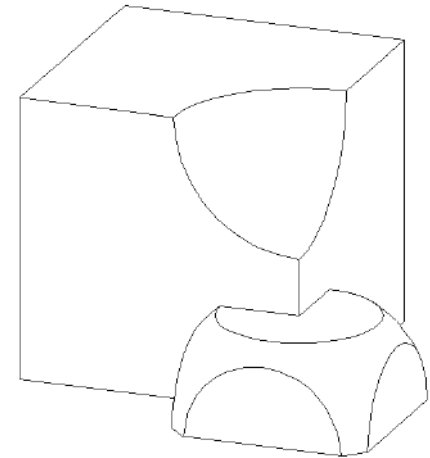
## KILLGROUP

**KILLGROUP** *g\_expr*

Удаляют из памяти тела, построенные указанной группой. После операции **KILLGROUP** группа становится пустой. Удаление тел производится автоматически по завершению интерпретации и при возврате из макровыводов. С точки зрения производительности эта команда должна использоваться, когда группа больше не нужна.

*Пример:*

```
GROUP "box"
  BRICK 1, 1, 1
ENDGROUP
GROUP "sphere"
  ADDZ 1
  SPHERE 0.45
  DEL 1
ENDGROUP
GROUP "semisphere"
  ELLIPS 0.45, 0.45
ENDGROUP
GROUP "brick"
  ADD -0.35, -0.35, 0
  BRICK 0.70, 0.70, 0.35
  DEL 1
ENDGROUP
result_1=SUBGROUP("box","sphere")      ! Вычитание "sphere" из "box"
result_2=ISECTGROUP("semisphere","brick") ! Пересечение "semisphere" с "brick"
result_3=ADDGROUP(result_1,result_2)    ! Объединение ранее созданных элементов
PLACEGROUP result_3
KILLGROUP "box"
KILLGROUP "sphere"
KILLGROUP "semisphere"
KILLGROUP "brick"
```



## SWEEPGROUP

**SWEEPGROUP** (g\_expr, x, y, z)

Возвращает группу, которая получается в результате поворота группы, заданной в качестве параметра, вдоль заданного направления. Команда действует только для моделей-тел.

**SWEEPGROUP (2)** (g\_expr, x, y, z)

Различие между SWEEPGROUP и SWEEPGROUP (2) заключается в том, что во втором варианте по отношению к вектору направления операции поворота применяется фактическая матрица преобразований по отношению к текущей системе координат. (В случае SWEEPGROUP текущее преобразование применяется к вектору направления дважды по отношению к глобальной системе координат.)

*Пример:*



```
GROUP "a"
  SPHERE 1
ENDGROUP
PLACEGROUP SWEEPGROUP ("a", 2, 0, 0)
```

## ИСПОЛЬЗОВАНИЕ ДВОИЧНОГО ФОРМАТА

### BINARY

**BINARY** mode [, section]

Это специальная команда для включения двоичных объектов в макрос GDL. Множество вершин, векторов, ребер, многоугольников тел и покрытий считываются из специального раздела файла библиотечного элемента. Они преобразуются в соответствии с текущими преобразованиями и объединяются в 3D-модель. Данные, содержащиеся в двоичном разделе, недоступны для редактирования пользователем.

mode: указывает использование реквизитов пера и покрытий.

0: имеют силу текущие установки PEN и MATERIAL.

1: текущие установки PEN и MATERIAL не действуют. Библиотечный элемент будет показан с учетом сохраненных определений цвета и покрытий. Внешний вид поверхности не изменяется.

- 2: используются сохраненные установки `PEN` и `MATERIAL`. Для неопределенных покрытий задаются текущие установки.
- 3: используются сохраненные установки `PEN` и `MATERIAL`. Для неопределенных покрытий задаются характеристики, хранимые по умолчанию.

section: индекс двоичного раздела, от 1 до 16.

Вы можете обращаться одновременно ко всем существующим двоичным разделам, используя нулевое значение индекса параметра section.

Из GDL могут быть сохранены только разделы с индексом 1. Команда `BINARY` с отсутствующим аргументом section также обращается к этому разделу. Остальные значения индексов разделов будут использоваться другими приложениями

Если открывается файл со структурой данных, отличающейся от ArchiCAD (например, DXF или ZOOM) то их 3D-описание будет преобразовано в двоичный формат.

Находясь в основном окне редактирования библиотечных элементов, Вы можете сохранить библиотечный элемент, выбрав команду *Сохранить как*. Если при этом Вы в последующем диалоговом окне отметите маркер *Сохранить в двоичном формате*, то текст GDL текущего библиотечного элемента будет заменен на двоичное описание.

**Совет:** сохранение в двоичном формате 3D-модели после выполнения операции создания 3D-разреза приведет к сохранению усеченной модели. Таким образом, Вы можете создавать фигуры с разрезами.

Вы можете запомнить библиотечный элемент в двоичном формате, только если уже была создана его 3D-модель.

Заменяя текстовое описание библиотечного элемента на двоичное, Вы существенно уменьшаете время построения его трехмерного изображения. С другой стороны, двоичное 3D-описание не является параметрическим и требует большего объема памяти на диске, чем алгоритмическая GDL-программа.



# ДВУМЕРНЫЕ ФИГУРЫ

В этой главе представлены команды, используемые для построения 2D-фигур простой формы, таких как линии, дуги, многоугольники и сплайн-кривые, а также для определения текстовых элементов в 2D. Здесь также рассматриваются способы оперирования двоичными данными в 2D, а также способы проецирования фигур, созданных 3D-скриптами, в 2D-виды, что гарантирует согласованность 3D- и 2D-представлений объектов. Наконец, представлены команды, позволяющие пользователям размещать графические элементы в списках, создаваемых в результате проведения различных вычислений.

## ЧЕРТЕЖНЫЕ ЭЛЕМЕНТЫ

### HOTSPOT2

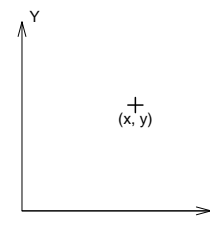
**HOTSPOT2**  $x, y$  [, unID [, paramReference, flags][, displayParam]]

unID: уникальный идентификатор узловой точки в 2D-скрипте. Он является полезным в том случае, когда в скрипте имеется переменное количество узловых точек.

paramReference: параметр, который можно редактировать с помощью этой узловой точки с использованием графического метода редактирования параметров узловой точки.

displayParam: параметр, который показывается в информационной панели при редактировании параметра paramReference. Можно использовать элементы массива.

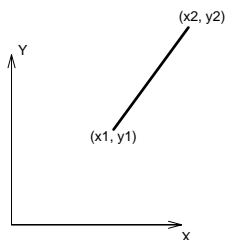
См. ["Графическое редактирование"](#) на стр. 135 для получения дополнительной информации об использовании HOTSPOT2.



### LINE2

**LINE2**  $x1, y1, x2, y2$

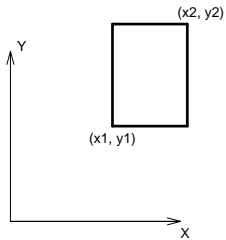
Определение отрезка, расположенного между двумя точками.



## RECT2

**RECT2**  $x1, y1, x2, y2$

Определение прямоугольника с помощью двух его диагональных вершин.



## POLY2

**POLY2**  $n, frame\_fill, x1, y1, \dots, xn, yn$

Открытый или замкнутый многоугольник с  $n$  вершинами.

*Ограничения на параметры:*

$n \geq 2$

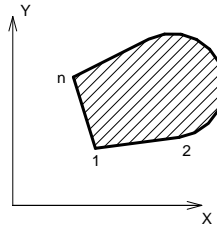
$frame\_fill = j1 + 2*j2 + 4*j3$

где  $j1, j2, j3$  могут принимать значения 0 или 1.

$j1$  (1): только контур.

$j2$  (2): только штриховка.

$j3$  (4): замкнуть открытый многоугольник.





## POLY2\_

**POLY2\_ n**, frame\_fill, x1, y1, s1, ... xn, yn, sn

Аналогично предложению POLY2 за исключением того, что любые ребра могут быть опущены. Если  $s_i = 0$ , то ребро, исходящее из вершины  $(x_i, y_i)$  будет опущено. Если  $s_i = 1$ , то ребро присутствует. Значение  $s_i = -1$  используется для определения отверстия. С помощью дополнительных кодов статуса в многоугольнике можно определить дуги и сегменты.

*Ограничения на параметры:*

$n \geq 2$

frame\_fill =  $j1 + 2*j2 + 4*j3 + 8*j4 + 32*j6 + 64*j7$

где  $j1, j2, j3$  могут принимать значения 0 или 1.

$j1$  (1): только контур.

$j2$  (2): только штриховка.

$j3$  (4): замкнуть открытый многоугольник.

$j4$  (8): ориентация локальной штриховки.

$j6$ : штриховка является штриховкой сечений (по умолчанию - штриховка чертежей).

$j7$ : штриховка является штриховкой поверхностей (только если  $j6 = 0$ , по умолчанию - штриховка чертежей)

*Значения статуса:*

$s = j1 + 16*j5 + 32*j6$

где  $j1, j5, j6$  могут принимать значения 0 или 1.

$j1$  (1): следующее ребро присутствует.

$j5$  (16): следующее ребро является внутренней линией (если 0, линия общего вида).

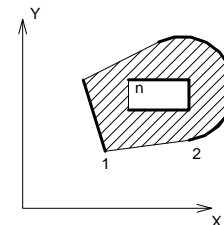
$j6$  (32): следующее ребро является линией контура (действует, только если не установлен  $j5$ ).

-1: конец контура.

Спецификация линии по умолчанию для линий POLY2\_ является 0 (линия общего вида), Предложение LINE\_PROPERTY не действует для ребер POLY2\_.

Дополнительные коды статуса позволяют создавать ребра и дуги в плоской ломаной с использованием специальных ограничений.

См. *“Дополнительные коды статусов”* на стр. 143 для получения дополнительной информации.



## POLY2\_A

```
POLY2_A n, frame_fill, fill_pen,
        x1, y1, s1, ..., xn, yn, sn
```

## POLY2\_B

```
POLY2_B n, frame_fill, fill_pen,
        fill_background_pen,
        x1, y1, s1, ..., xn, yn, sn
```

Расширенный вариант предложения POLY2\_ с дополнительными параметрами пера штриховки, пера фона штриховки. Все остальные параметры аналогичны определенным в предложении POLY2\_. Дополнительные коды статуса позволяют создавать ребра и дуги в плоской ломаной с использованием специальных ограничений.

См. *“Дополнительные коды статусов” на стр. 143* для получения дополнительной информации.

## POLY2\_B{2}

```
POLY2_B{2} n, frame_fill, fill_pen,
            fill_background_pen,
            fillOrigoX, fillOrigoY,
            fillAngle,
            x1, y1, s1, ..., xn, yn, sn
```

Расширенный вариант предложения POLY2\_ с дополнительными параметрами пера штриховки, пера фона штриховки, начала размещения штриховки и ее направления.

```
frame_fill = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 312*j6 + 64*j7
```

где j1, j2, j3, j4 j5, j6, j7 могут принимать значения 0 или 1.

j1 (1): только контур.

j2 (2): только штриховка.

j3 (4): замкнуть открытый многоугольник.

j4 (8): ориентация локальной штриховки.

j5 (16): глобальное начало штриховки (действует, только если установлен j4).

j6 (32): штриховка принадлежит категории штриховки сечений (в противовес j7, если не установлен, то принимается категория штриховки чертежей).

j7 (64): штриховка принадлежит категории штриховки поверхностей (в противовес j6; если не установлен, то принимается категория штриховки чертежей).

Дополнительные коды статуса позволяют создавать ребра и дуги в плоской ломаной с использованием специальных ограничений.

См. *“Дополнительные коды статусов” на стр. 143* для получения дополнительной информации.

## POLY2\_B{3}

```
POLY2_B{3} n, frame_fill, fill_pen,
            fill_background_pen,
            fillOrigoX, fillOrigoY,
            mxx, mxy, myx, myy, x1, y1, s1, ..., xn, yn, sn
```

Расширенный вариант предложения POLY2\_ с дополнительными возможностями начало размещения штриховки и ее направление с использованием матрицы.

$\text{frame\_fill} = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8$

где  $j1, j2, j3, j4, j5, j6, j7, j8$  могут принимать значения 0 или 1.

$j1-j7$ : аналогичны предыдущим командам POLY2\_.

$j8$  (128): использовать наклонную штриховку.

$mxx, mxy, myx, myy$ : если  $j8$  установлен, то эта матрица определяет ориентацию штриховки.

Дополнительные коды статуса позволяют создавать ребра и дуги в плоской ломаной с использованием специальных ограничений.

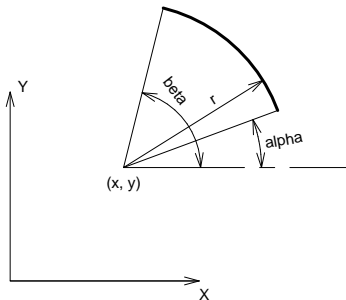
См. *“Дополнительные коды статусов”* на стр. 143 для получения дополнительной информации.

## ARC2

```
ARC2 x, y, r, alpha, beta
```

Дуга с центром в точке  $(x, y)$ , радиусом  $r$  и ограниченная углами  $\alpha$  и  $\beta$ .

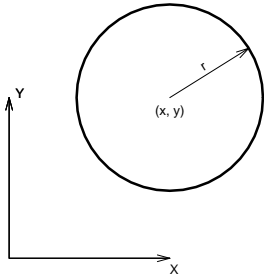
$\alpha$  и  $\beta$  измеряются в градусах.



## CIRCLE2

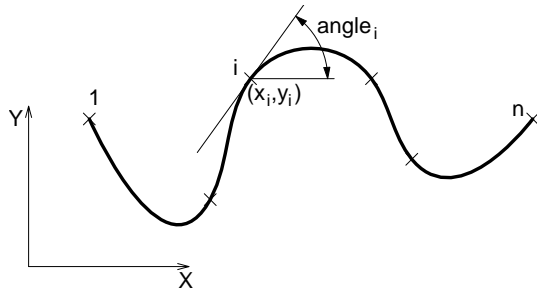
**CIRCLE2**  $x, y, r$

Окружность с центром в точке  $(x, y)$  и радиусом  $r$ .



## SPLINE2

**SPLINE2**  $n, status, x1, y1,$   
 $angle1, \dots, xn, yn, anglen$



*Ограничения на параметры:*

$n \geq 2$

Сплайн-кривая, построенная по  $n$  контрольным точкам. Касательная сплайн-кривой в контрольной точке  $(x_i, y_i)$  определяется параметром  $angle_i$ , угол касательной с осью  $x$  измеряется в градусах.

*Значения статуса:*

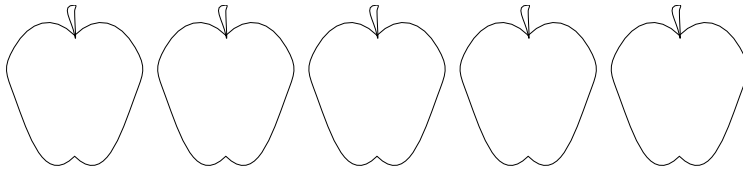
0: по умолчанию.

1: замкнутая сплайн-кривая. Первая и последняя вершины соединяются, замыкая, таким образом, сплайн-кривую.

2: автоматически сглаживаемая сплайн-кривая. При создании сплайн-кривой не учитываются значения параметров angle между первой и последней вершинами. Используется внутренний алгоритм сглаживания.

*Примеры.*

```
SPLINE2 5, 2,
         0, 0, 60,
         1, 2, 30,
         1.5, 1.5, -30,
         3, 4, 45,
         4, 3, -45
```



```
n = 5
FOR I = 1 TO n
  SPLINE2 4, 0,
          0.0, 2.0, 135.0,
          -1.0, 1.8, 240.0,
          -1.0, 1.0, 290.0,
          0.0, 0.0, 45.0
  MUL2 -1.0, 1.0
  SPLINE2 4, 0,
          0.0, 2.0, 135.0,
          -1.0, 1.8, 240.0,
          -1.0, 1.0, 290.0,
          0.0, 0.0, 45.0
  DEL 1
  SPLINE2 4, 0,
          0.0, 2.0, 100.0,
          0.0, 2.5, 0.0,
          0.0, 2.4, 270.0,
          0.0, 2.0, 270.0
  ADD2 2.5, 0
NEXT I
```

## SPLINE2A

```
SPLINE2A n, status, x1, y1, angle1, length_previous1, length_next1,
...
xn, yn, anglen, length_previousn,
length_nextn
```

Обобщение предложения SPLINE2 (сплайн-кривая Безье). Безье). В силу своей сложности используется в основном при автоматической генерации 2D-скрипта.

Для получения дополнительной информации см. “Построение чертежей /Линии/ Построение сплайн-кривых” в главе “Создание документации” справки ArchiCAD 11.

Коды статуса:

0: по умолчанию.

1: замкнутая сплайн-кривая. Первая и последняя вершины соединяются, замыкая, таким образом, сплайн-кривую.

2: автоматически сглаживаемая сплайн-кривая. При создании сплайн-кривой не учитываются значения параметров  $angle_i$ ,  $length\_previous_i$  и  $length\_next_i$  между первой и последней вершинами. Используется внутренний алгоритм сглаживания.

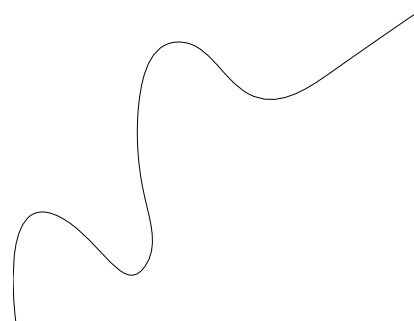
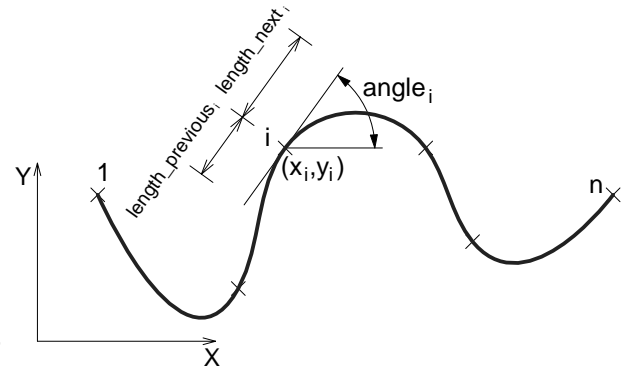
$x_i, y_i$ : координаты вершин.

$length\_previous_i, length\_next_i$ : длины манипуляторов касательных с обеих сторон вершин.

$angle_i$ : угол, определяющий направление касательной.

Пример:

```
SPLINE2A 9, 2,
0.0, 0.0, 0.0, 0.0, 0.0,
0.7, 1.5, 15, 0.9, 1.0,
1.9, 0.8, 72, 0.8, 0.3,
1.9, 1.8, 100, 0.3, 0.4,
1.8, 3.1, 85, 0.4, 0.5,
2.4, 4.1, 352, 0.4, 0.4,
3.5, 3.3, 338, 0.4, 0.4,
4.7, 3.7, 36, 0.4, 0.8,
6.0, 4.6, 0, 0.0, 0.0
```



## PICTURE2

**PICTURE2** expression, a, b, mask

### PICTURE2{2}

**PICTURE2{2}** expression, a, b, mask

Может использоваться на плоскости по аналогии с предложением PICTURE в пространстве. В отличие от 3D, в 2D-рисунках значения маски не используются.

Значение вычисленного выражения expression строкового типа обозначает имя файла, а числового типа - индекс рисунка, хранящегося в библиотечном элементе. Индекс 0 указывает на рисунок образца, воспроизводимый в окошке просмотра при выборе библиотечного элемента. Для PICTURE2{2} mask = 1 означает, что абсолютно белые пиксели являются прозрачными. Другие рисунки могут быть сохранены в библиотечных элементах только при сохранении всего проекта или выбранных элементов, содержащих рисунки в формате объекта GDL.

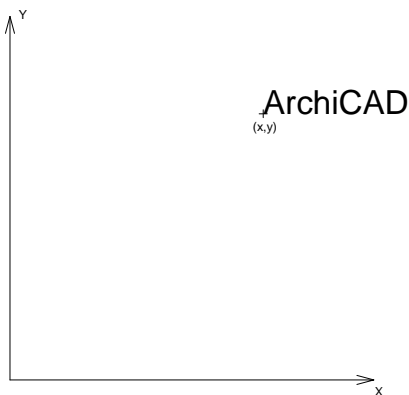
## ТЕКСТОВЫЕ ЭЛЕМЕНТЫ

### TEXT2

**TEXT2** x, y, expression

Значение вычисленного выражения expression числового или строкового типа выводится в установленном ранее стиле, начиная с позиции с координатами x,y.

См. также команды “[SET] STYLE” на стр. 157 и “DEFINE STYLE” на стр. 176.



## RICHTEXT2

**RICHTEXT2** *x*, *y*, *textblock\_name*

Приводит к размещению ранее определенного TEXTBLOCK.

Для получения дополнительной информации см. *“Определение текстового блока”* на стр. 178.

*x*, *y*: координаты X-Y размещения обогащенного текста (richtext).

*textblock\_name*: имя ранее определенного TEXTBLOCK.

## ИСПОЛЬЗОВАНИЕ ДВУМЕРНЫХ ДАННЫХ В ДВОИЧНОМ ФОРМАТЕ

### FRAGMENT2

**FRAGMENT2** *fragment\_index*,  
*use\_current\_attributes\_flag*

Фрагмент с указанным индексом *fragment\_index* выводится в окно полного 2D-вида с учетом текущих преобразований.

*use\_current\_attributes\_flag*: указывает, будут ли использоваться текущие реквизиты.

0: фрагмент выводится с учетом цвета, типа линии и типа штриховки, определенных для него.

1: вместо цвета, типа линии и типа штриховки, определенных для фрагмента, используются текущие установки скрипта.

### FRAGMENT2

**FRAGMENT2** ALL,*use\_current\_attributes\_flag*

Все фрагменты выводятся в окно полного 2D-вида с учетом текущих преобразований.

*use\_current\_attributes\_flag*: указывает, будут ли использоваться текущие реквизиты.

0: фрагменты выводятся с учетом цвета, типа линий и типа штриховки, определенных для них.

1: вместо цвета, типа линии и типа штриховки, определенных для фрагментов, используются текущие установки скрипта.



## ПРОЕКЦИИ ТРЕХМЕРНЫХ ФИГУР НА ПЛОСКОСТЬ

### PROJECT2

**PROJECT2** projection\_code, angle, method

### PROJECT2{2}

**PROJECT2{2}** projection\_code, angle, method [,backgroundColor, fillOrigoX, fillOrigoY, filldirection]

Создает проекцию 3D-скрипта в том же библиотечном элементе и добавляет построенные линии в параметрический 2D-символ. Второй вариант, **PROJECT2{2}**, вместе с ранее определенной командой **SET FILL**, позволяет пользователю управлять фоном штриховки, а также началом и направлением размещения результирующего чертежа из 2D-скрипта.

projection\_code: тип проекции.

3: вид сверху.

4: вид сбоку.

6: фронтальная аксонометрия.

7: изометрия.

8: монометрия.

9: диметрия.

-3: вид снизу.

-6: фронтальная аксонометрия, вид снизу.

-7: изометрия, вид снизу.

-8: монометрия, вид снизу.

-9: диаметрия, вид снизу.

angle: азимут, заданный в диалоговом окне *Определение 3D-проекции*.

method: выбранный метод визуализации.

1: каркасная модель.

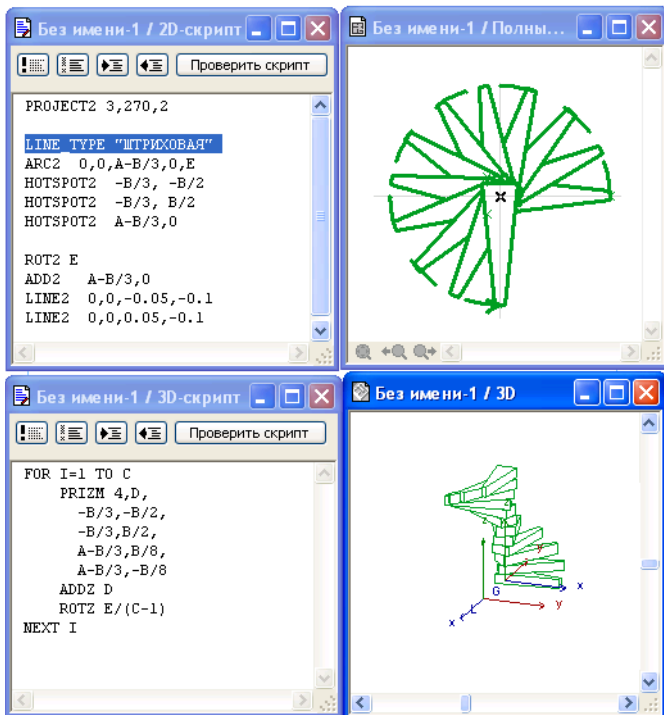
2: аналитический метод с удалением невидимых линий.

3: с раскраской и тенями.

- 16: дополнительный модификатор: векторная штриховка (действует только в методах визуализации с удалением невидимых линий и с раскраской и тенями).
- 32: дополнительный модификатор: использовать текущие реквизиты вместо реквизитов из 3D (действует только в методе визуализации с раскраской и тенями).
- 64: дополнительный модификатор: ориентация локальной штриховки. (действует только в методе визуализации с раскраской и тенями).
- 128: дополнительный модификатор: все линии являются внутренними (действует только вместе с 32). По умолчанию - линия общего вида.
- 256: дополнительный модификатор: все линии являются линиями контура (действует только вместе с 32, если не установлен 128). По умолчанию - линия общего вида.
- 512: дополнительный модификатор: вся штриховка является штриховкой сечения (действует только вместе с 32). По умолчанию берется штриховка чертежей.
- 1024: дополнительный модификатор: вся штриховка является штриховкой поверхности (действует только вместе с 32, если не установлен 512). По умолчанию выбирается штриховка чертежа.
- BackgroundColor: цвет фона штриховки.
- fillOrigoX: координата X начала штриховки.
- fillOrigoY: координата Y начала штриховки.
- filldirection: угол направления штриховки.

**Примечание:** Для PROJECT2 { 2 } действует SET FILL.

Пример:



**Примечание о совместимости:** При использовании PROJECT2 при не установленном бите 32 и установленном бите 3 (с раскраской и тенями) для параметра method, модель будет отсекается с помощью CUTPOLYA без установленного бита статуса 2 (создание многоугольников сечения), в результате чего реквизиты многоугольника сечения могут отличаться. Многоугольники сечения будут создаваться с реквизитами, определенными в команде SECT\_FILL, расположенной в 3D-скрипте.

## PROJECT2{3}

```
PROJECT2{3} projection_code, angle, method , parts[, backgroundColor, fillOrigoX, fillOrigoY,
  fillldirection][[, ]
PARAMETERS name1=value1 , ... namen=valuen]
```

Создает проекцию 3D-скрипта в том же самом библиотечном элементе и добавляет созданные линии в параметрический 2D-символ. Второй вариант, PROJECT2{2}, вместе с предыдущей командой SET FILL, позволяет пользователю управлять фоном штриховки, а также началом и направлением размещения результирующего чертежа из 2D-скрипта. Третий вариант, PROJECT2{3}, обладает дополнительной возможностью определять, какая именно часть спроецированной модели необходима, а также предоставляет возможность независимо управлять реквизитами отсекаемой и оставляемой частями модели, включая и тип линии. Вы также можете создавать проекцию с конкретными параметрами, указываемыми в этой команде.

method: выбранный метод визуализации.

1: каркасная модель.

2: аналитический метод с удалением невидимы линий.

3: с раскраской и тенями.

16: дополнительный модификатор: векторная штриховка (действует только в методах визуализации с удалением невидимых линий и с раскраской и тенями).

32: дополнительный модификатор: использовать текущие реквизиты вместо реквизитов из 3D (действует только в методе визуализации с раскраской и тенями).

64: дополнительный модификатор: ориентация локальной штриховки. (действует только в методе визуализации с раскраской и тенями).

128: дополнительный модификатор: все линии являются внутренними (действует только вместе с 32). По умолчанию - линия общего вида.

256: дополнительный модификатор: все линии являются линиями контура (действует только вместе с 32, если не установлен 128). По умолчанию - линия общего вида.

512: дополнительный модификатор: вся штриховка является штриховкой сечения (действует только вместе с 32). По умолчанию выбирается штриховка чертежей.

1024: дополнительный модификатор: вся штриховка является штриховкой поверхности (действует только вместе с 32, если не установлен 512). По умолчанию выбирается штриховка чертежа.

2048: дополнительный модификатор: модификаторы 16, 32, 64, 128, 256, 512, 1024 и параметры реквизита штриховки являются действительными только для просматриваемой части проекции. По умолчанию они действуют для всех частей.

4096: дополнительный модификатор: модификаторы 16, 32, 64, 128, 256, 512, 1024 и параметры реквизита штриховки являются действительными только для отсекаемой части проекции. По умолчанию они действуют для всех частей.

8192: дополнительный модификатор: штриховка сечения является наклонной

**Известное ограничение:** линии отсекаемой части не могут трактоваться по отдельности, только все вместе взятые линии могут быть установлены в качестве внешних линий или линий контура.

parts: определяет создаваемые части. Значение 15 означает все части.

$part = j1 + 2*j2 + 4*j3 + 8*j4$

где  $j1, j2, j3, j4$  могут принимать значения 0 или 1. Значения  $j1, j2, j3, j4$  указывают, будут ли соответствующие части проецируемой модели представлены (1) или опущены (0).

$j1$ : отсекаемые многоугольники(со значениями по умолчанию для штриховки, определенной согласно `SECT_FILL`) (действует только в модели с раскраской и тенями).

$j2$ : ребра отсекаемых многоугольников.

$j3$ : показываемые многоугольники.

$j4$ : ребра показываемых многоугольников.

## ВКЛЮЧЕНИЕ ЧЕРТЕЖЕЙ В СМЕТУ ЭЛЕМЕНТОВ

Команды этой группы действуют только в том случае, когда смета элементов создается в ArchiCAD.

Если библиотечный элемент относится к категории специальных то есть является библиотечным элементом спецификации, и некоторым образом связан с обычным библиотечным элементом (объектом, дверью, окном или источником света), размещенным на плане этажа, то включение ниже приведенных предложений в 2D-скрипт библиотечного элемента спецификации означает ссылку на 2D- или 3D-описание библиотечного элемента. Это виртуальная ссылка, которая "материализуется" в процессе составления сметы с использованием 2D- или 3D-скрипта выводимого в текущий момент элемента.

## DRAWING2

**DRAWING2** [expression]

В зависимости от значения expression, создает чертеж библиотечного элемента (expression = 0, по умолчанию) или выносную надпись этого элемента (expression = 1), ассоциируемую с объектом спецификаций, содержащим эту команду.

## DRAWING3

**DRAWING3** projection\_code, angle, method

## DRAWING3{2}

**DRAWING3{2}** projection\_code, angle, method [,backgroundColor, origoX, origoY, fillldirection]

Как и предложение PROJECT2, создает проекцию 3D-скрипта библиотечного элемента, связанного с библиотечным элементом спецификации, содержащим данную команду. Все параметры аналогичны параметрам предложений PROJECT2 и PROJECT2{2}.

*В DRAWING3{2} имеется новый параметр method:*

3: с раскраской и тенями,

32: использовать текущие реквизиты, а не реквизиты из 3D,

64: ориентация локальной штриховки.

## DRAWING3{3}

**DRAWING3{3}** projection\_code, angle, method , parts[, backgroundColor, fillOrigoX, fillOrigoY, fillldirection][[,] PARAMETERS name1=value1 , ... namen=valuen]

Как и предложение PROJECT2, создает проекцию 3D-скрипта библиотечного элемента, связанного с библиотечным элементом спецификации, содержащим данную команду. Все параметры аналогичны параметрам предложений PROJECT2, PROJECT2{2} и PROJECT2{3}.

*В DRAWING3{3} имеется новый параметр method:*

2048: дополнительный модификатор: модификаторы 16, 32, 64, 128, 256, 512, 1024 и параметры реквизита штриховки являются действительными только для просматриваемой части проекции. По умолчанию они действуют для всех частей.

4096: дополнительный модификатор: модификаторы 16, 32, 64, 128, 256, 512, 1024 и параметры реквизита штриховки являются действительными только для отсекаемой части проекции. По умолчанию они действуют для всех частей.

8192: дополнительный модификатор: штриховка сечения является наклонной.

---

# ГРАФИЧЕСКОЕ РЕДАКТИРОВАНИЕ

*Интерактивное графическое редактирование параметров GDL линейного или углового типа с помощью узловых точек.*

## КОМАНДА РЕДАКТИРОВАНИЯ С ПОМОЩЬЮ УЗЛОВЫХ ТОЧЕК

### HOTSPOT

**HOTSPOT** *x*, *y*, *z* [, *unID* [, *paramReference*, *flags*] [, *displayParam*]]

**HOTSPOT2** *x*, *y* [, *unID* [, *paramReference*, *flags*][, *displayParam*]]

*unID*: уникальный идентификатор узловой точки в пределах библиотечного элемента, в котором эта узловая точка определена.

*paramReference*: параметр, который может редактироваться этой узловой точкой с использованием графического метода редактирования параметров на основе узловых точек.

*displayParam*: параметр, который должен показываться в информационной панели при редактировании параметра *paramReference*. Могут также передаваться элементы массива.

*Примеры допустимых аргументов:*

*D*, *Arr*[5], *Arr*[2\*I+3][D+1] и т.д.

*flags*:тип узловой точки + реквизит узловой точки.

*Тип:*

- 1: редактирование линейного типа, базисная узловая точка;
- 2: редактирование линейного типа, перемещаемая узловая точка;
- 3: редактирование линейного типа, ссылочная узловая точка (всегда спрятана);
- 4: редактирование углового типа, базисная узловая точка;
- 5: редактирование углового типа, перемещаемая узловая точка;
- 6: редактирование углового типа, центр угла (всегда спрятана);
- 7: редактирование углового типа, ссылочная узловая точка (всегда спрятана).

*Реквизитом может быть комбинация следующих значений или ноль:*

128: спрятать узловую точку имеет смысл для типов: 1,2,4,5);

256: редактируемая базисная узловая точка (для типов: 1,4);

512: изменить направление измерения угла в 2D (для типа 6).

Для редактирования параметра линейного типа следует определить три узловые точки с типами 1, 2 и 3. Положительное направление линии редактирования задается вектором, идущим от ссылочной узловой точки к базисной узловой точке. Перемещаемая узловая точка должна быть размещена вдоль этой линии на расстоянии, определяемым значением сопутствующего параметра, измеряемым от базисной узловой точки.

Для редактирования параметра углового типа следует определить четыре узловые точки с типами 4, 5, 6 и 7. Плоскость угла располагается перпендикулярно вектору, идущему от узловой точки центра и к ссылочной узловой точке. Положительным направлением измерения угла является против часовой стрелки при направлении взгляда на плоскость угла от ссылочной узловой точки. В 2D плоскость уже задана, поэтому ссылочная узловая точка игнорируется, а положительное направление измерения угла берется по умолчанию против часовой стрелки. Это направление можно изменить с помощью параметра 512, задаваемого для центральной узловой точки (тип 6). Для согласованности вектора, идущие от центральной узловой точки к перемещаемой и базисной узловым точкам, должны быть перпендикулярны вектору, идущему от центра к ссылочной узловой точке. Перемещаемая узловая точка должна быть размещена под углом, определяемым сопутствующим параметром, измеряемым от базисной узловой точки вокруг центральной узловой точки.

Если для редактирования одного и того же параметра определено несколько наборов узловых точек, то узловые точки группируются для выполнения команд этих узловых точек. Если для базисной узловой точки установлен редактируемый реквизит, то пользователь может редактировать такой параметр путем перемещения базисной узловой точки. Так как предполагается, что базисная узловая точка является фиксированной внутри координатной рамки, охватывающей объект (то есть ее расположение не зависит от параметра, с которым она связана), то перемещение или поворот всего объекта производится вместе с этой базисной точкой. (При изменении значения параметра место расположения перемещаемого параметра не изменяется.)

Два или три набора узловых точек линейного типа могут быть объединены, чтобы можно было редактировать два или три параметра при одном перемещении. Если объединены два набора узловых точек, то перемещение узловой точки теперь ограничивается не линией, а плоскостью, определяемой двумя линиями из каждого из наборов узловых точек редактирования линейных размеров. В 3D, объединение трех наборов узловых точек редактирования линейных размеров позволяет расположить узловую точку в любом месте пространства. Две линии не должны быть параллельны друг другу, а три линии не должны располагаться в одной плоскости. Комбинированная операция редактирования параметра выполняется, если в месте расположения выбранной точки имеются две или три редактируемые узловые точки (перемещаемые или редактируемые базисные) с различными ассоциируемыми с ними параметрами. Если параметры предназначены для совместного редактирования, то базисная и ссылочная узловые точки не являются фиксированными в координатной рамке объекта; это означает, что они должны перемещаться по мере изменения значения другого параметра.

*См. иллюстрацию и пример 2.*



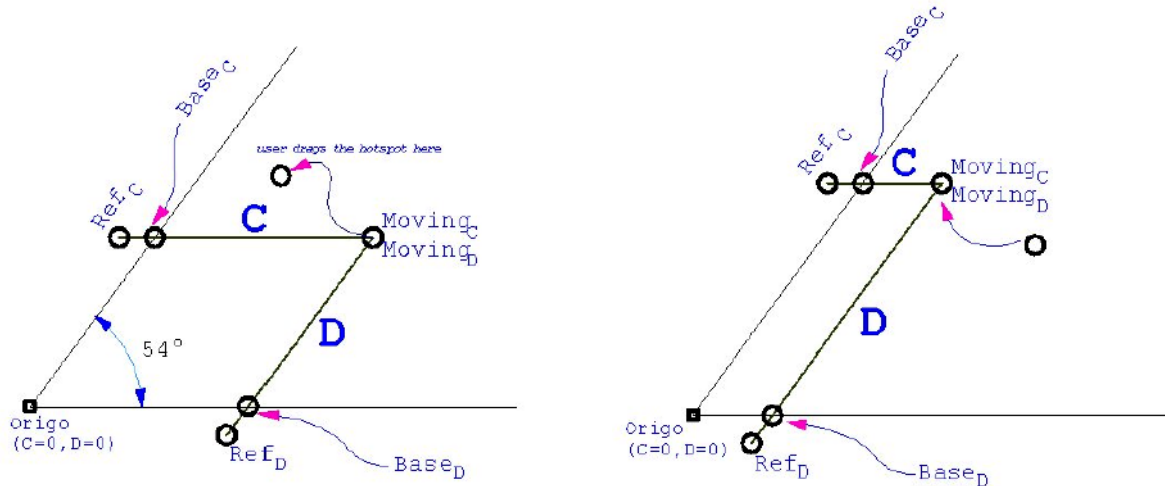
Пример 1: редактирование угла в 2D:

```

LINE2 0, 0, A, 0
LINE2 0, 0, A*cos(angle), A*sin(angle)
ARC2 0, 0, 0.75*A, 0, angle
HOTSPOT2 0, 0, 1, angle, 6
HOTSPOT2 0.9*A, 0, 2, angle, 4
HOTSPOT2 0.9*A*cos(angle), 0.9*A*sin(angle), 3, angle, 5

```

Пример 2, совместное редактирование линейного типа с помощью двух параметров:



```

RECT2 0, 0, A, B
RECT2 0, 0, sideX, sideY
HOTSPOT2 sideX, 0, 1, sideY, 1
HOTSPOT2 sideX, -0.1, 2, sideY, 3
HOTSPOT2 sideX, sideY, 3, sideY, 2
HOTSPOT2 0, sideY, 4, sideX, 1
HOTSPOT2 -0.1, sideY, 5, sideX, 3
HOTSPOT2 sideX, sideY, 6, sideX, 2

```

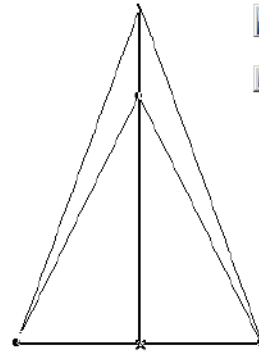
Пример 3, простое редактирование линейного типа с помощью одного параметра:

2D-СКРИПТ:

```
HOTSPOT2 -1, 0, 1
HOTSPOT2 1, 0, 2
HOTSPOT2 0, 0, 3, corner_y, 1+128
HOTSPOT2 0, -1, 4, corner_y, 3
HOTSPOT2 0, corner_y, 5, corner_y, 2
LINE2 -1, 0, 1, 0
LINE2 -1, 0, 0, corner_y
LINE2 1, 0, 0, corner_y
```

3D-СКРИПТ:

```
HOTSPOT -1, 0, 0, 1
HOTSPOT -1, 0, 0.5, 2
HOTSPOT 1, 0, 0, 3
HOTSPOT 1, 0, 0.5, 4
HOTSPOT 0, 0, 0, 5, corner_y, 1+128
HOTSPOT 0, -1, 0, 6, corner_y, 3
HOTSPOT 0, corner_y, 0, 7, corner_y, 2
HOTSPOT 0, 0, 0.5, 8, corner_y, 1+128
HOTSPOT 0, -1, 0.5, 9, corner_y, 3
HOTSPOT 0, corner_y, 0.5, 10, corner_y, 2
PRISM_ 4, 0.5,
    -1, 0, 15,
    1, 0, 15,
    0, corner_y, 15,
    -1, 0, -1
```



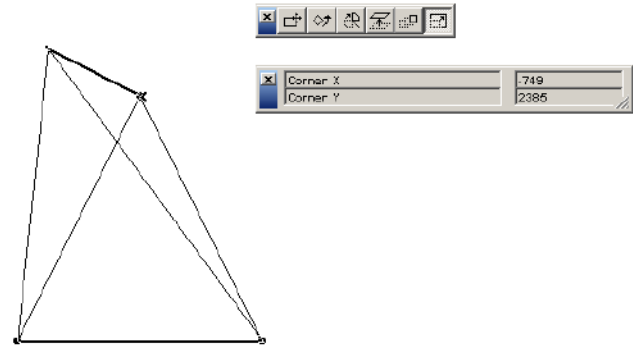
Пример 4: совместное редактирование линейного типа с помощью двух параметров:

2D-СКРИПТ:

```
HOTSPOT2 -1, 0, 1
HOTSPOT2 1, 0, 2
HOTSPOT2 corner_x, 0, 3, corner_y, 1+128
HOTSPOT2 corner_x, -1, 4, corner_y, 3
HOTSPOT2 corner_x, corner_y, 5, corner_y, 2
HOTSPOT2 0, corner_y, 3, corner_x, 1+128
HOTSPOT2 -1, corner_y, 4, corner_x, 3
HOTSPOT2 corner_x, corner_y, 5, corner_x, 2
LINE2 -1, 0, 1, 0
LINE2 -1, 0, corner_x, corner_y
LINE2 1, 0, corner_x, corner_y
```

3D-СКРИПТ:

```
HOTSPOT -1, 0, 0, 1
HOTSPOT -1, 0, 0.5, 2
HOTSPOT 1, 0, 0, 3
HOTSPOT 1, 0, 0.5, 4
HOTSPOT corner_x, 0, 0, 5, corner_y, 1+128
HOTSPOT corner_x, -1, 0, 6, corner_y, 3
HOTSPOT corner_x, corner_y, 0, 7, corner_y, 2
HOTSPOT 0, corner_y, 0, 8, corner_x, 1+128
HOTSPOT -1, corner_y, 0, 9, corner_x, 3
HOTSPOT corner_x, corner_y, 0, 10, corner_x, 2
HOTSPOT corner_x, 0, 0.5, 11, corner_y, 1+128
HOTSPOT corner_x, -1, 0.5, 12, corner_y, 3
HOTSPOT corner_x, corner_y, 0.5, 13, corner_y, 2
HOTSPOT 0, corner_y, 0.5, 14, corner_x, 1+128
HOTSPOT -1, corner_y, 0.5, 15, corner_x, 3
HOTSPOT corner_x, corner_y, 0.5, 16, corner_x, 2
PRISM_ 4, 0.5,
-1, 0, 15,
1, 0, 15,
corner_x, corner_y, 15,
-1, 0, -1
```



## **HOTLINE2**

**HOTLINE2**  $x_1, y_1, x_2, y_2$

Определение линии статуса между двумя точками.

## **HOTARC2**

**HOTARC2**  $x, y, r, startangle, endangle$

Определение дуги статуса с центом в точке  $(x, y)$ , начальным углом  $startangle$ , конечным углом  $endangle$  и радиусом  $r$ .

---

# КОДЫ СТАТУСОВ

*Коды статусов, приводимые в этой главе, позволяют пользователям создавать отрезки и дуги плоских ломаных линий с помощью специальных ограничений.*

Плоские ломаные линии со значениями статуса их вершин используются в следующих фигурах GDL:

POLY , PLANE , PRISM , CPRISM , BPRISM , FPRISM\_ , HPRISM\_ , SPRISM\_ , SLAB\_ , CSLAB\_ , CROOF\_ , EXTRUDE , PYRAMID , REVOLVE , SWEEP , TUBE , TUBEA

Коды статуса позволяют:

- управлять видимостью ребер плоской ломаной линии;
- определять отверстия в ломаных линиях;
- управлять видимостью боковых ребер и поверхностей;
- создавать отрезки и дуги ломаной линии.

## СИНТАКСИС КОДОВ СТАТУСОВ

$s_i$  - это двоичное целое число (между 0 и 127) или -1.

$s = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7$  [+ a\_code]

где  $j_1, j_2, j_3, j_4, j_5, j_6, j_7$  могут принимать значения 0 или 1.

Значения  $j_1, j_2, j_3, j_4$  определяют, присутствуют ли (1) или опущены (0) соответствующие вершины и стороны:

$j_1$ : нижнее горизонтальное ребро;

$j_2$ : вертикальное ребро;

$j_3$ : верхнее горизонтальное ребро;

$j_4$ : боковое ребро;

$j_5$ : горизонтальное ребро при удалении линии (только для фигур PRISM\_);

$j_6$ : вертикальное ребро при удалении линии (только для фигур PRISM\_);

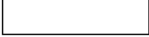






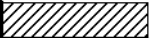








$j_7$ : специальное дополнительное значение статуса, являющееся действительным только когда  $j_2=1$ , которое контролирует видимость текущего вертикального ребра в зависимости от направления взгляда:

$j_2=0$ : вертикальное ребро всегда видимо;

$j_2=1$  и  $j_7=1$ : вертикальное ребро видимо только в том случае, когда оно является контуром, наблюдаемым из текущего направления взгляда;

$j_2=1$  и  $j_7=0$ : вертикальное ребро всегда видимо.

Возможные значения статуса (жирные линии обозначают видимые ребра):

<i>невидимая поверхность</i>	<i>видимая поверхность</i>
0 	8 
1 	9 
2 	10 
3 	11 
4 	12 
5 	13 
6 	14 
7 	15 

`a_code`: дополнительный код статуса (факультативный), который позволяет создавать отрезки и дуги в ломаной линии.

`si=-1` используется для непосредственного определения отверстий в призме. Этот код указывает на завершение определения контура и начало определения отверстия в контуре. Он также используется для указания конца контура отверстия и начала определения другого отверстия. Координаты перед этим значением должны быть идентичными координатам первой точки контура/отверстия. Если Вы используете значение маски `-1`, то последним значением маски в списке параметров должно быть `-1`, что указывает на завершении определения последнего отверстия.

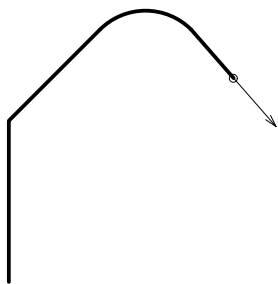
Отверстия должны быть разъединенными и их внутреннее пересечение в многоугольнике запрещено с целью правильного отображения результатов построения теней и реалистических изображений.

## ДОПОЛНИТЕЛЬНЫЕ КОДЫ СТАТУСОВ

Приводимые далее дополнительные коды статусов позволяют создавать отрезки и дуги ломаной линии с использованием специальных ограничений. Они определяют отрезок или дугу, продолжающую ломаную в данной точке. Исходный код статуса действует только в том случае, если он указан (после дополнительного кода статуса располагаются символы “+s”).

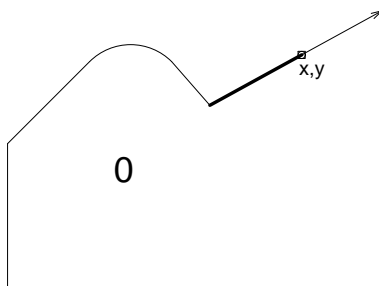
**Примечание:** Аппроксимация дуг управляется директивами, описанными в главе “Реквизиты”. При построении дуг с помощью предложения POLY2\_, реальную гладкую дугу можно получить только в случае аппроксимации не менее, чем 8 сегментами; в противном случае все дуги будут сегментированными.

### Предыдущее состояние ломаной: заданы текущее положение и направляющая



### Отрезок по абсолютным координатам второй точки

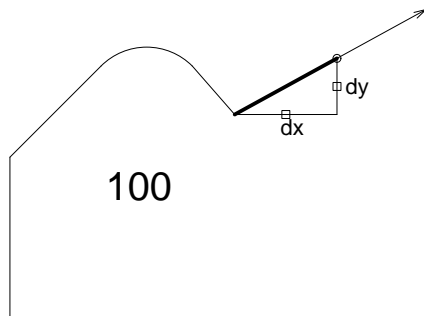
$x, y, s$   
где  $0 < s < 100$



## Отрезок по относительным координатам второй точки

$dx, dy, 100+s,$

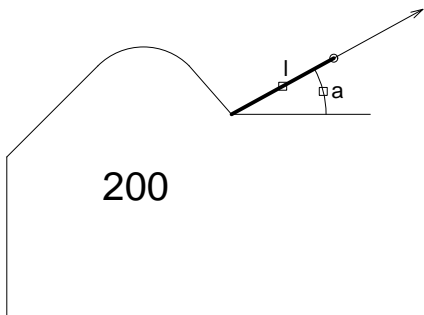
где  $0 < s < 100$



## Отрезок по длине и направлению

$l, a, 200+s,$

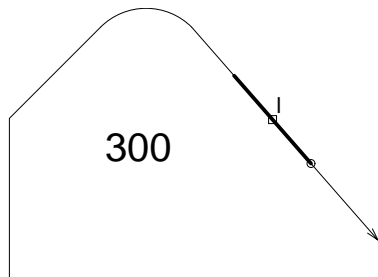
где  $0 < s < 100$





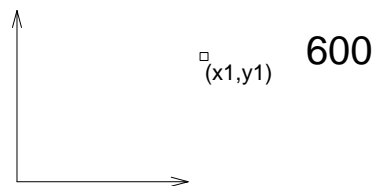
## Отрезок по длине вдоль направляющей

$l, 0, 300+s,$   
где  $0 < s < 100$



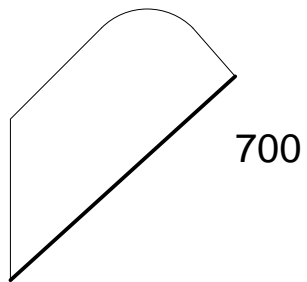
## Установить начальную точку

$x1, y1, 600,$



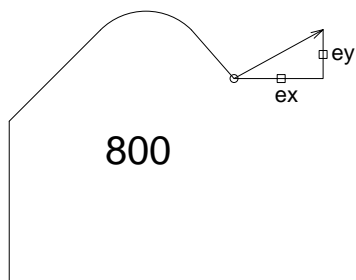
## Замкнуть ломаную

0, 0, 700,



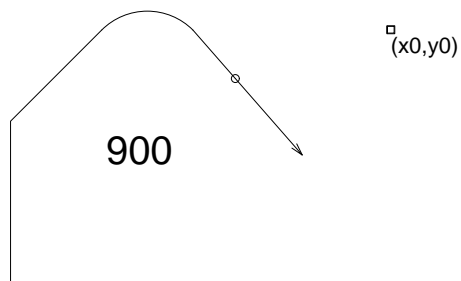
## Установить направляющую

ex, ey, 800,



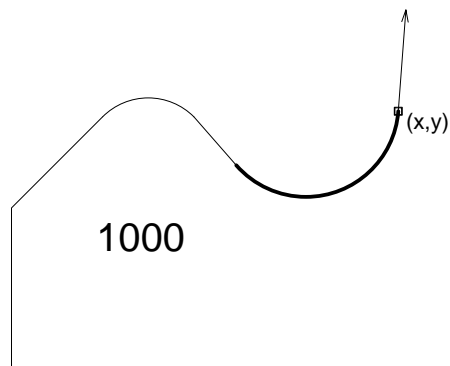
## Установить точку центра

$x_0, y_0, 900,$



## Дуга, касающаяся направляющей и оканчивающаяся в точке

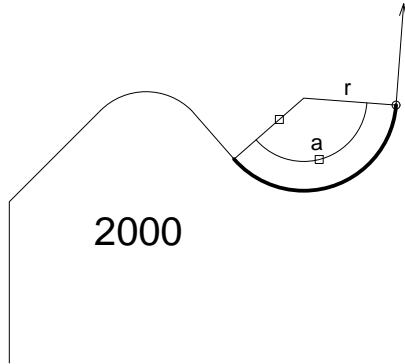
$x, y, 1000+s,$   
где  $0 < s < 100$



## Дуга, касающаяся направляющей и имеющая заданные радиус и угол

$r, a, 2000+s,$

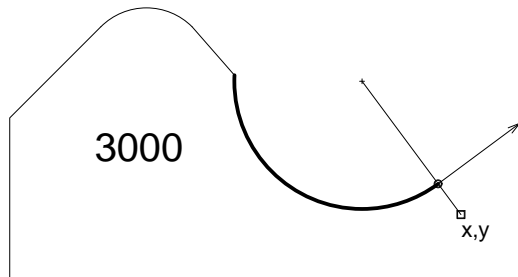
где  $0 < s < 100$



## Дуга по точке центра и точке на радиусе конца дуги

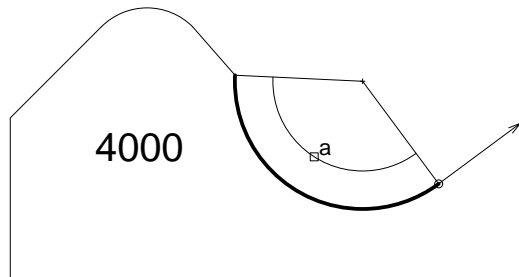
$x, y, 3000+s,$

где  $0 < s < 100$



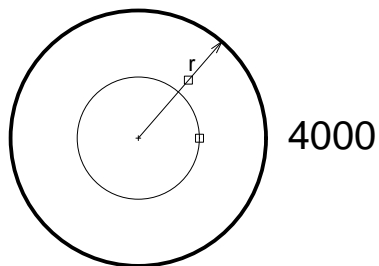
## Дуга по точке центра и углу

$0, a, 4000+s,$   
где  $0 < s < 100$



## Окружность по ее центру и радиусу

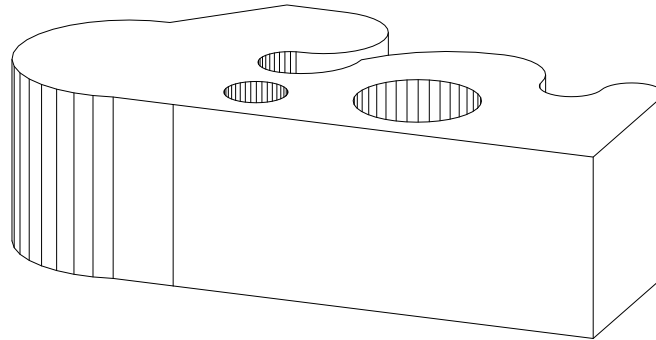
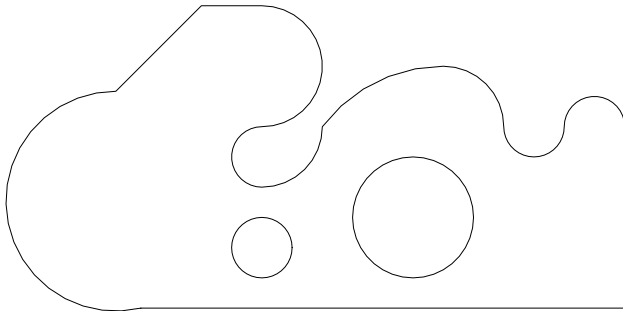
$r, 360, 4000+s,$   
где  $0 < s < 100$



В данном случае влияние статуса  $s$  распространяется на всю окружность.

Все углы измеряются в градусах. Неиспользуемые координаты, которые приведены со значениями 0 (для кодов 300, 700, 4000), могут иметь произвольные значения.

## Примеры;



```

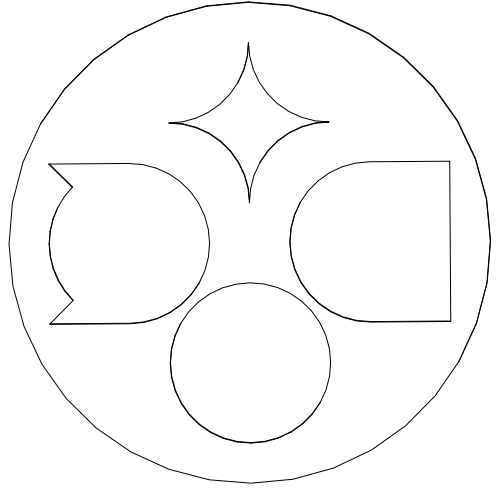
EXTRUDE 21, 0, 0, 3, 1+2+4+16+32,
0, 0, 0,
7, 0, 0,
7, 3, 1,
6, 3, 1000,      ! Дуга, касающаяся направляющей и оканчивающаяся в точке
5, 3, 1001,      ! Дуга, касающаяся направляющей и оканчивающаяся в точке
1, 90, 2000,     ! Дуга, касающаяся направляющей и имеющая заданные радиус и угол
2, 3, 1001,      ! Дуга, касающаяся направляющей и оканчивающаяся в точке
1, 3, 900,       ! Установить точку центра
1, 2, 3000,      ! Дуга по точке центра и точке на радиусе конца дуги
1, 2.5, 900,     ! Установить точку центра
0, -180, 4001,   ! Дуга по точке центра и углу
1, 5, 1000,      ! Дуга, касающаяся направляющей и оканчивающаяся в точке
-1, 0, 100,      ! Отрезок по относительным координатам
2, 225, 200,     ! Отрезок по длине и направлению
-1, 0, 800,      ! Установить направляющую
-1, 0, 1000,     ! Дуга, касающаяся направляющей и оканчивающаяся в точке
0, 0, -1,        ! Конец контура
1, 1, 900,       ! Установить точку центр
0.5, 360, 4000,  ! Окружность по ее центру и радиусу
3.5, 1.5, 900,   ! Установить точку центр
1, 360, 4001     ! Окружность по ее центру и радиусу

```

```

EXTRUDE 2+5+10+10+2, 0, 0, 3, 1+2+4+16+32,
0, 0, 900,
3, 360, 4001,
2.5, -1, 0,
2.5, 1, 0,
1.5, 1, 1,
1.5, -1, 1001,
2.5, -1, -1,
0, 2.5, 600,
0, -1, 800,
1, 1.5, 1001,
-1, 0, 800,
0, 0.5, 1001,
0, 1, 800,
-1, 1.5, 1001,
1, 0, 800,
0, 2.5, 1001,
0, 2.5, 700,
-1.5, 0, 900,
-2.5, 0, 600,
-2.5, 1, 3000,
-2.5, 1, 0,
-1.5, 1, 0,
-1.5, -1, 1001,
-2.5, -1, 0,
SQRT(2)-1, 45, 200,
-2.5, 0, 3000,
-2.5, 0, 700,
0, -1.5, 900,
1, 360, 4000

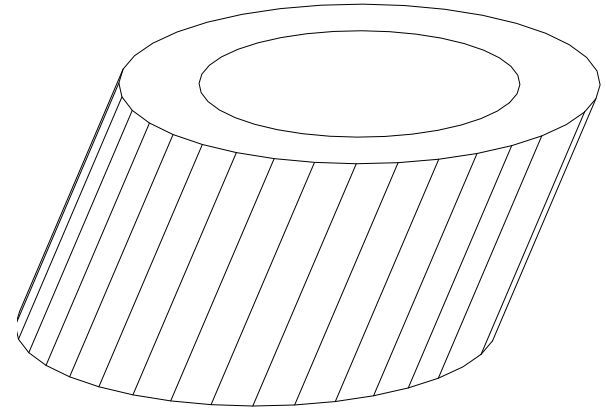
```



```

EXTRUDE 3, 1, 1, 3, 1+2+4+16+32,
0, 0, 900,
3, 360, 4001,
2, 360, 4000

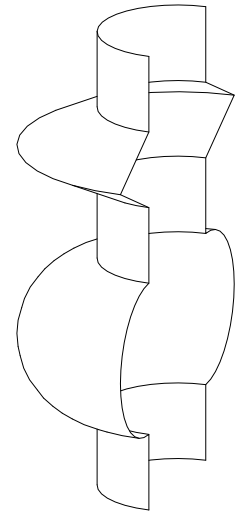
```



```

ROTY -90
REVOLVE 9, 180, 16+32,
7, 1, 0,
6, 1, 0,
5.5, 2, 0,
5, 1, 0,
4, 1, 0,
3, 1, 900, ! Установить точку центра
0, 180, 4001, ! Дуга по точке центра и углу
2, 1, 0,
1, 1, 0

```





---

# РЕКВИЗИТЫ

*В первой части этой главы представлены директивы, оказывающие влияние на интерпретация последующих предложений GDL. Директивы могут устанавливать сглаженность цилиндрических элементов, указывать способ представления 3D-видов или производить присвоение реквизитов (цвет, покрытие, стиль текста, и т.д.) для последующих фигур. Во второй части главы приводятся определения реквизитов. В этом случае предоставляется возможность приписать конкретным объектам специальные характеристики покрытий, текстуры, образцов штриховки, типов линий и стилей текста, которые отсутствуют в текущем наборе реквизитов Вашего проекта.*

## ДИРЕКТИВЫ

Директивы оказывают влияние на интерпретацию последующих предложений GDL. Они имеют силу до поступления следующей аналогичной директивы или до конца скрипта. Вызываемый скрипт наследует текущие установки директив. Все производимые в нем изменения установок директив имеют локальный характер. Это означает, что после выхода из скрипта восстанавливаются те установки директив, которые были сделаны до обращения к нему.

### Директивы, используемые в 3D- и 2D-скриптах

#### [LET]

[**LET**] varnam = n

Присвоение значения переменной. Фраза LET не является обязательной. Переменной varnam присваивается значение n.

## RADIUS

**RADIUS** radius\_min, radius\_max

Устанавливает сглаженность цилиндрических элементов и дуг в ломаной.

*Окружность радиусом  $r$  представляется:*

- если  $r < \text{radius\_min}$ , то шестиугольником,
- если  $r \geq \text{radius\_max}$ , то 36-угольником,
- если  $\text{radius\_min} < r < \text{radius\_max}$ , то  $(6+30*(r-\text{radius\_min})/(\text{radius\_max}-\text{radius\_min}))$ -угольником.

Дуга представляется пропорционально ее размеру.

После предложения RADIUS, предыдущие предложения RESOL и TOLER теряют свою силу.

*Ограничения на параметры:*

$r_{\min} \leq r_{\max}$

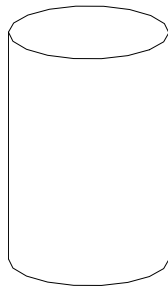
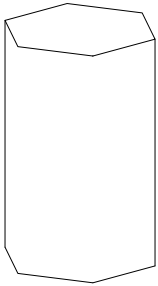
*Примеры:*

RADIUS 1.1, 1.15

RADIUS 0.9, 1.15

CYLIND 3.0, 1.0

CYLIND 3.0, 1.0



## RESOL

**RESOL** n

Устанавливает сглаженность цилиндрических элементов и дуг в ломаных. Окружности представляются правильными n-угольниками.

Дуга представляется пропорционально ее размеру.

После предложения RESOL предыдущие предложения RADIUS и TOLER теряют свою силу.

*Ограничения на параметры:*

n >= 3

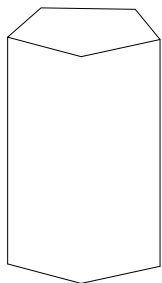
*По умолчанию:*

RESOL 36

*Примеры:*

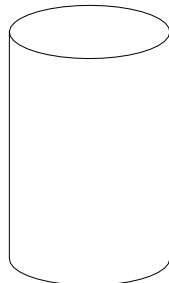
RESOL 5

CYLIND 3.0, 1.0



RESOL 36

CYLIND 3.0, 1.0



## TOLER

### TOLER d

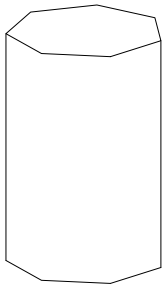
Устанавливает сглаженность цилиндрических элементов и дуг в ломаных. Погрешность аппроксимации окружности (дуги), то есть наибольшее расстояние между предполагаемой дугой и аппроксимирующей ее хордой, будет не более d.

После предложения TOLER предыдущие предложения RADIUS и RESOL теряют свою силу.

*Примеры:*

```
TOLER 0.1
CYLIND 3.0, 1.0
```

```
TOLER 0.01
CYLIND 3.0, 1.0
```



**Примечание:** Директивы RADIUS, RESOL и TOLER устанавливают сглаженность цилиндрических 3D-элементов (CIRCLE, ARC, CYLIND, SPHERE, ELLIPS, CONE, ARMC, ARME, ELBOW, REVOLVE) и дуг в 2D-ломаных с использованием криволинейных ребер.

См. *“Дополнительные коды статусов”* на стр. 143.

## PEN

### PEN n

Установка цвета пера.

*Ограничения на параметры:*

$$0 < n \leq 255$$

*По умолчанию:*

```
PEN 1
```

если в скрипте нет предложения PEN.

(Для библиотечных элементов ArchiCAD берет значения по умолчанию из установок параметров соответствующих библиотечных элементов. Если скрипт ссылается на несуществующий индекс, то установленным значением по умолчанию является PEN 1).

## LINE\_PROPERTY

**LINE\_PROPERTY** expr

Определяет характеристики всех последующих создаваемых линий в 2D-скрипте (команды RECT2, LINE2, ARC2, CIRCLE2, SPLINE2, SPLINE2A, POLY2, FRAGMENT2) вплоть до следующего предложения LINE\_PROPERTY. По умолчанию - линия общего вида.

*Возможные значения для expr:*

0: все линии являются общего вида;

1: все линии являются внутренними;

2: все линии являются контуром.

## [SET] STYLE

**[SET] STYLE** name\_string

**[SET] STYLE** index

Все последующий тексты будут иметь данный стиль до поступления нового предложения SET STYLE.

Индекс - это константа, которая ссылается на стек стилей, являющийся составной частью внутренней структуры данных ArchiCAD (отрицательные индексы указывают на те индексы внутренней структуре покрытий, которые были ранее определены в скрипте GDL). Этот стек изменяется при анализе GDL, а также может быть изменен из программы. Использование индекса вместо названия стиля рекомендуется только совместно с функцией IND, описанной ранее.

*По умолчанию:*

## SET STYLE 0

(текущий шрифт, размер 5 мм, точка привязки равна 1, гарнитура обычная), если в скрипте нет предложения SET STYLE.

## Директивы, используемые только в 3D-скриптах

### MODEL

**MODEL** WIRE

**MODEL** SURFACE

**MODEL** SOLID

Устанавливает способ представления создаваемых объектов.

**MODEL WIRE:** каркасная модель. Отсутствуют поверхности и внутренняя часть объектов. Объекты являются прозрачными.

MODEL SURFACE, MODEL SOLID: построение сечений основывается на пересечении соседних поверхностей. Обе модели создают одну и ту же внутреннюю структуру трехмерных данных. Объекты не являются прозрачными.

Отличие этих двух моделей обнаруживается только в том случае, когда часть объекта отсекается какой-либо плоскостью:

MODEL SURFACE: так как объект пустотелый, то видна его внутренняя часть,

MODEL SOLID: в плоскости сечения создается новая поверхность.

*По умолчанию:*

MODEL SOLID

Для иллюстрации этих моделей рассмотрим следующие три куба:

MODEL WIRE

BLOCK 3,2,1

ADDY 4

MODEL SURFACE

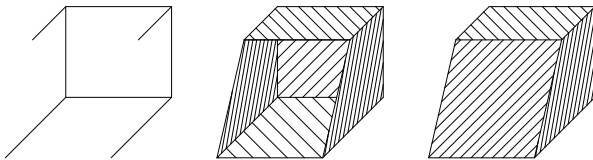
BLOCK 3,2,1

ADDY 4

MODEL SOLID

BLOCK 3,2,1

После построения их среза получим следующее:



## [SET] MATERIAL

[SET] MATERIAL name\_string

[SET] MATERIAL index

Все последующие поверхности будут иметь указанное покрытие до поступления новой команды MATERIAL. Поверхности тел

BPRISM\_, CPRISM\_, FPRISM\_, HPRISM\_  
SPRISM\_, CSLAB\_, CWALL\_, BWALL\_, XWALL\_,  
CROOF\_, MASS

являются исключением из этого правила.

Индекс - это константа, которая ссылается на стек покрытий -составной части внутренней структуры данных ArchiCAD

(отрицательные индексы указывают на те индексы внутренней структуре покрытий, которые были ранее определены в скрипте

GDL). Этот стек изменяется при анализе GDL, а также может быть изменен из программы. Использование индекса вместо названия покрытия рекомендуется только совместно с функцией `IND`, описанной ранее.

Индекс 0 имеет специальное значение: для поверхностей используется цвет текущего пера и они имеют матовый тип покрытия.

*По умолчанию:*

`MATERIAL 0`

если в скрипте нет предложения `MATERIAL`.

(Для библиотечных элементов ArchiCAD берет значения по умолчанию из установок параметров соответствующих элементов.

Если скрипт ссылается на несуществующий индекс, то значением по умолчанию является `MATERIAL 0`).

*См. также описание функции `IND` в “Разное” > “Запросы” на стр. 245.*

## SECT\_FILL

```
SECT_FILL fill, fill_background_pen,
          fill_pen, contour_pen
```

или

```
SECT_ATTRS fill, fill_background_pen,
            fill_pen, contour_pen [, line_type]
```

Определяет реквизиты, используемые в отсекаемой части 3D-элементов в окне разреза/фасада и в команде `PROJECT2 { 3 }` (для достижения совместимости предыдущие версии команды `PROJECT2` не затрагиваются).

`fill`: имя штриховки или номер индекса.

`fill_background_pen`: номер пера фона штриховки.

`fill_pen`: номер цвета пера штриховки.

`contour_pen`: номер цвета пера контура штриховки.

`line_type`: тип линии ребер многоугольников.

## SHADOW

**SHADOW** keyword\_1[, keyword\_2]

Управляет отбрасыванием теней элементами при фотосъемке и при векторном построении теней.

keyword\_1: ON, AUTO или OFF.

keyword\_2: ON или OFF.

ON: все последующие элементы всегда отбрасывают тень.

OFF: последующие элементы никогда не будут отбрасывать тень.

AUTO: отбрасывание теней управляется автоматически.

- Установка SHADOW OFF для скрытых частей объекта экономит память и время обработки.
- Установка SHADOW ON обеспечит отбрасывание теней даже самыми незначительными элементами объекта.

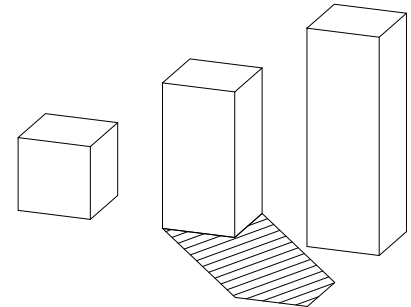
Необязательный второй параметр keyword управляет построением теней на поверхностях.

- SHADOW keyword\_1, OFF деактивирует векторное построение теней для последующих поверхностей.
- SHADOW keyword\_1, ON включает векторное построение теней.

*По умолчанию:*

SHADOW AUTO

```
SHADOW OFF
BRICK 1, 1, 1
ADDX 2
SHADOW ON
BRICK 1, 1, 2
ADDX 2
SHADOW OFF
BRICK 1, 1, 3
```





## Директивы, используемые только в 2D-скриптах

### DRAWINDEX

**DRAWINDEX** number

Определяет порядок построения элементов 2D-скрипта. Элементы с меньшим drawindex чертятся раньше.

*Ограничения на параметры:*

$0 < \text{number} \leq 50$

(В текущей версии GDL для DRAWINDEX являются допустимыми только значения 0, 20, 30, 40 и 50. Другие значения округляются до этих величин.)

Если директива DRAWINDEX отсутствует, порядок черчения элементов по умолчанию следующий:

- 1 рисунки;
- 2 штриховка;
- 3 линии;
- 4 текстовые элементы.

### [SET] FILL

**[SET] FILL** name\_string

**[SET] FILL** index

Все последующие 2D-многоугольники будут заштрихованы согласно этому образцу до поступления нового предложения SET FILL.

Индекс - это константа, которая ссылается на стек образцов штриховки, являющийся составной частью внутренней структуры данных ArchiCAD. Этот стек изменяется при анализе GDL, а также может быть изменен из программы. Использование индекса вместо названия образца штриховки рекомендуется только совместно с функцией IND, описанной ранее.

*По умолчанию:*

SET FILL 0

то есть ее отсутствие (пустая штриховка), если в скрипте нет предложения SET FILL.

*См. также описание функции IND в “Разное” > “Запросы” на стр. 245.*

## [SET] LINE\_TYPE

[SET] LINE\_TYPE name\_string

[SET] LINE\_TYPE index

Все последующие 2D-линии (линии, дуги, многоугольники) будут иметь этот тип до поступления нового предложения SET LINE\_TYPE.

Индекс - это константа, которая ссылается на стек типов линий, являющийся составной частью внутренней структуры данных ArchiCAD. Этот стек изменяется при анализе GDL, а также может быть изменен из программы. Использование индекса вместо названия образца штриховки рекомендуется только совместно с функцией IND, описанной ранее.

*По умолчанию:*

```
SET LINE_TYPE 1
```

то есть 'Сплошная', если в скрипте нет предложения SET LINE\_TYPE.

*См. также описание функции IND в “Разное” > “Запросы” на стр. 245.*

## ОПРЕДЕЛЕНИЕ РЕКВИЗИТОВ

В ArchiCAD реквизиты задаются в диалоговых окнах определения параметров покрытия, штриховки и типа линии. К этим реквизитам плана этажа можно получить доступ из любого скрипта GDL.

Кроме того, реквизиты могут также определяться прямо в GDL-скрипте. Имеется два варианта такого определения.

- Определение реквизитов в скрипте MASTER\_GDL. Скрипт MASTER\_GDL выполняется при загрузке в оперативную память содержащей его библиотеки. Реквизиты MASTER\_GDL объединяются с реквизитами плана этажа; при этом реквизиты ArchiCAD с совпадающими именами не заменяются. Сразу после загрузки MASTER\_GDL определенные в нем реквизиты становятся доступными любым скриптам.
- Определение реквизитов в библиотечных элементах. Определенные таким образом покрытия и текстура могут использоваться в самом скрипте и в скриптах нижнего уровня. Определенные и используемые в 2D-скрипте штриховка и типы линий ведут себя так же, как если бы они были определены в скрипте MASTER\_GDL.

Команда *Проверить GDL-скрипты* в диалоговом окне библиотечного элемента помогает проверить корректно ли определены параметры покрытия, штриховки, типа линии или стиля.

Если покрытие, штриховка, тип линии или стиль в трехмерном представлении библиотечного элемента отличаются от предполагаемого, и при этом не выдано ни одного сообщения об ошибке, это означает, что, скорее всего, некорректно заданы значения одного или более параметров. Команда *Проверить GDL-скрипты* поможет в этом случае обнаружить эти параметры, выдав более детальные сообщения.

## Определение покрытия

### DEFINE MATERIAL

**DEFINE MATERIAL** name type, parameter1,  
parameter2, ... parametern

**Примечание:** Эта команда может содержать определение дополнительных данных.

См. *“Дополнительные данные” на стр. 179* для получения дополнительной информации.

Любой скрипт GDL может содержать определения покрытий до первого использования их названий. Покрытие, определенное таким образом, может использоваться только для 3D-элементов в самом скрипте и в скриптах нижнего уровня.

**name:** название покрытия.

**type:** тип покрытия. Фактическое количество (n) параметров, которые определяют покрытие, отличается в зависимости от типа покрытия. Смысл параметров и их допустимые значения приведены в последующих примерах.

0: общее определение, n=16

1: простое определение, n=9 (дополнительные параметры являются константами или вычисляются из заданных значений).

2-7: предварительно определенные типы покрытий, n=3

Три задаваемых значения - это компоненты RGB цвета поверхности. Другие параметры являются константами или вычисляются из цвета.

2: матовое.

3: металл.

4: пластик.

5: стекло.

6: люминесцентное.

7: постоянное.

10: общее определение с параметром штриховки, n=17.

11: простое определение с параметром штриховки, n=10.

12-17: предварительно определенные типы покрытий с параметром штриховки, n=4.

20: общее определение с параметрами штриховки, индекса цвета штриховки и индекса текстуры, n=19

21: простое определение с параметрами штриховки, индекса цвета штриховки и индекса текстуры, n=12

22-27: предварительно определенные типы покрытий с параметрами штриховки, индекса цвета штриховки и индекса текстуры, n=6

**Специальные значения для типов 20-27:**

Если номер пера равен нулю, векторная штриховка создается с использованием активного пера.

Нулевое значение индекса текстуры позволяет Вам определять покрытия без векторной штриховки или текстуры.

*Примеры.*

```
DEFINE MATERIAL "вода" 0,  
    0.5284, 0.5989, 0.6167,  
!   цвет покрытия RGB [0.0..1.0]  
    1.0, 0.5, 0.5, 0.9,  
!   коэффициенты рассеянного света, диффузии, зеркального отражения и прозрачности [0.0..1.0]  
    2.0,  
!   сфокусированность бликов [0.0..100.0]  
    1,  
!   анизотропный эффект прозрачности [0.0..4.0]  
    0.5284, 0.5989, 0.6167,  
!   цвет зеркальных бликов RGB [0.0..1.0]  
    0, 0, 0,  
!   цвет люминесценции RGB [0.0..1.0]  
    0.0  
!   ослабление интенсивности люминесцентности [0.0..65.5]  
DEFINE MATERIAL "асфальт" 1,  
    0.1995, 0.2023, 0.2418,  
!   цвет покрытия RGB [0.0..1.0]  
    1.0, 1.0, 0.0, 0.0,  
!   коэффициенты рассеянного света, диффузии, зеркального отражения и прозрачности [0.0..1.0]  
    0,  
!   сфокусированность бликов [0..100]  
    0  
!   анизотропный эффект прозрачности [0..4]  
DEFINE MATERIAL "красное матовое" 2,  
    1.0, 0.0, 0.0  
!   цвет покрытия RGB [0.0..1.0]  
DEFINE MATERIAL "кирпич красный" 10,  
    0.878294, 0.398199, 0.109468,  
    0.58, 0.85, 0.0, 0.0,  
    0,  
    0.0,  
    0.878401, 0.513481, 0.412253,  
    0.0, 0.0, 0.0,  
    0,
```

```

IND(FILL, "кирпич обыкновенный")
! индекс штриховки
DEFINE MATERIAL "кирпич желтый+*" 20,
  1, 1, 0,
! цвет покрытия RGB [0.0 .. 1.0]
  0.58, 0.85, 0, 0,
! коэффициенты рассеянного света, диффузии, зеркального отражения и прозрачности [0.0.. 1.0]
  0,
! сфокусированность бликов [0.0 .. 100.0]
  0,
! анизотропный эффект прозрачности [0.0 .. 4.0]
  0.878401, 0.513481, 0.412253,
! цвет зеркальных бликов RGB [0.0 .. 1.0]
  0, 0, 0,
! цвет люминесценции RGB [0.0 .. 1.0]
  0,
! ослабление интенсивности люминесценции [0.0 .. 65.5]
  IND(FILL, "кирпичный блок 25x75"), 61,
  IND(TEXTURE, "кирпич")
! индекс штриховки, индекс цвета, индекс текстуры

```

## DEFINE MATERIAL BASED\_ON

```

DEFINE MATERIAL name [,] BASED_ON orig_name [,] PARAMETERS name1 = expr1 [, ...][[,]
ADDITIONAL_DATA name1 = expr1 [, ...]]

```

Определение покрытия базируется на существующем покрытии. Указанные параметры исходного покрытия будут заменяться новыми значениями, другие параметры остаются нетронутыми. Использование этой команды без фактических параметров приведет к тому, что получается такое же покрытие, но под другим именем. Значения параметров покрытия могут быть получены с использованием функции *“REQUEST{2} (“Material\_info”, name\_or\_index, param\_name, value\_or\_values)”*.

*orig\_name*: название исходного покрытия (имя существующего, ранее определенного в GDL или на плане этажа, покрытия).

*name1*: название параметра покрытия, заменяемого новым значением. Имена, соответствующие параметрам определения покрытия, следующие:

```

gs_mat_surface_r, gs_mat_surface_g, gs_mat_surface_b (цвет поверхности RGB [0.0..1.0])
gs_mat_ambient (коэффициент рассеянного света [0.0..1.0])
gs_mat_diffuse (коэффициент диффузии [0.0..1.0])
gs_mat_specular (коэффициент зеркальных бликов [0.0..1.0])
gs_mat_transparent (коэффициент прозрачности [0.0..1.0])
gs_mat_shining (сфокусированность бликов [0.0..100.0])

```

gs\_mat\_transp\_att (анизотропный эффект прозрачности [0.0..4.0])  
 gs\_mat\_specular\_r, gs\_mat\_specular\_g, gs\_mat\_specular\_b (цвет зеркальных бликов RGB [0.0..1.0])  
 gs\_mat\_emission\_r, gs\_mat\_emission\_g, gs\_mat\_emission\_b (цвет люминесценции RGB [0.0..1.0])  
 gs\_mat\_emission\_att (ослабление интенсивности люминесцентности [0.0..65.5])  
 gs\_mat\_fill\_ind (индекс штриховки)  
 gs\_mat\_fillcolor\_ind (индекс цвета штриховки)  
 gs\_mat\_texture\_ind (индекс текстуры)

expr<sub>i</sub>: новое значение, заменяемое указанный параметр покрытия. Диапазон значений такой же, как и для определения покрытия.

*Пример:*

```

n = REQUEST{2} ("Material_info", "Brick-Face", "gs_mat_emission_rgb ", em_r, em_g, em_b)
em_r = em_r + (1 - em_r) / 3
em_g = em_g + (1 - em_g) / 3
em_b = em_b + (1 - em_b) / 3
DEFINE MATERIAL "Brick-Face light" [,] BASED_ON "Brick-Face"
PARAMETERS gs_mat_emission_r = em_r, gs_mat_emission_g = em_g, gs_mat_emission_b = em_b

SET MATERIAL "Brick-Face"
BRICK a, b, zzyzx
ADDX a
SET MATERIAL "Brick-Face light"
BRICK a, b, zzyzx
  
```

## DEFINE TEXTURE

**DEFINE TEXTURE** name expression, x, y, mask, angle

Любой GDL-скрипт может содержать определение текстуры до первого использования ее названия. Текстура, определенная таким образом, может использоваться только в самом скрипте и в скриптах нижнего уровня.

name: имя текстуры

expression: рисунок, связанный с текстурой. Выражение строкового типа указывает на имя файла, а выражение числового типа указывает на индекс рисунка, хранимого в библиотечном элементе. Индекс 0 - это специальное значение, которое ссылается на рисунок предварительного просмотра библиотечного элемента.

x: логическая ширина текстуры.

y: логическая высота текстуры.

mask:  $j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8 + 256*j9$   
 где  $j1, j2, j3, j4, j5, j6, j7, j8, j9$  могут принимать значения 0 или 1.

Управление альфа-каналом (j1... j6):

j1: альфа-канал изменяет прозрачность текстуры.

j2: изменение тиснения или смещение нормали к поверхности.

При нанесении тиснения используется альфа-канал, чтобы определить амплитуду колебания нормали к поверхности.

j3: альфа-канал изменяет цвет диффузии текстуры.

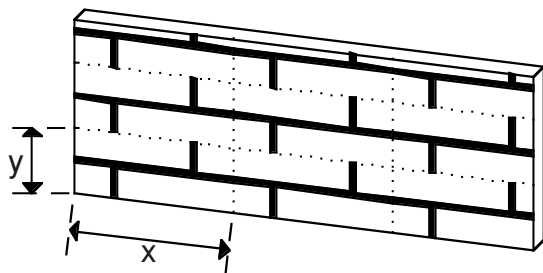
j4: альфа-канал изменяет цвет зеркальных бликов текстуры.

j5: альфа-канал изменяет цвет рассеянного света текстуры.

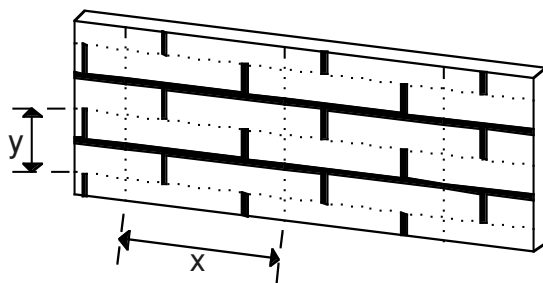
j6: альфа-канал изменяет цвет поверхности текстуры.

Управление соединением (j7... j9):

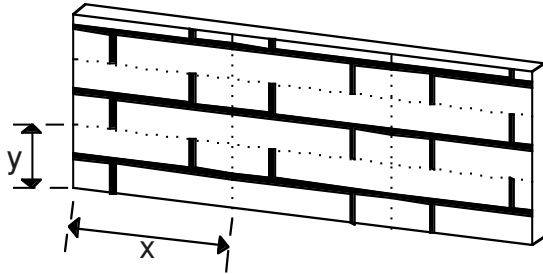
Если значение параметра равно нулю, то выбирается обычный режим:



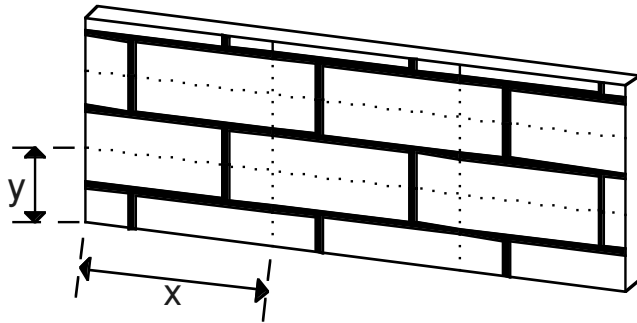
j7: текстура будет сдвинута случайным образом.



j8: зеркальное отражение по оси x.



j9: зеркальное отражение по оси y.



angle: угол поворота.

*Пример:*

```
DEFINE TEXTURE "Кирпич" "Кирпич.PICT", 1.35, 0.3, 256+128, 35.0
```

## Определение штриховки

### DEFINE FILL

```
DEFINE FILL name [[,] FILLTYPES_MASK fill_types,] pattern1, pattern2, pattern3, pattern4,
  pattern5, pattern6, pattern7, pattern8,
  spacing, angle, n,
  frequency1, direction1, offset_x1, offset_y1, m1,
  length11, ... length1m,
  ...
```



```

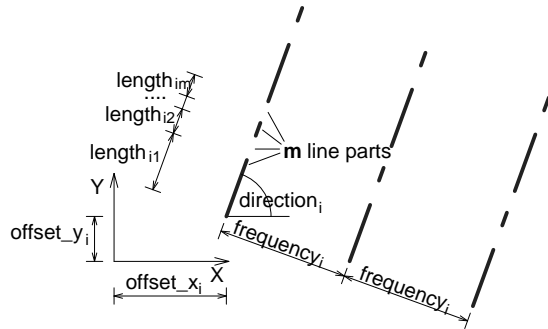
frequencyn, directionn, offset_xn,
lengthn1, ... lengthnm

```

**Примечание:** Эта команда может содержать определение дополнительных данных.

См. *“Дополнительные данные”* на стр. 179 для получения дополнительной информации.

Любой скрипт GDL может содержать определения штриховок до первого использования их названий. Штриховка, определенная таким образом, может использоваться только в самом скрипте и в скриптах нижнего уровня.



name: название образца штриховки.

fill\_types = j1 + 2 \* j2 + 4 \* j3.

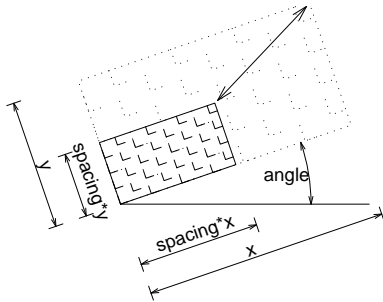
j1: штриховка сечений.

j2: штриховка поверхностей.

j3: штриховка чертежей.

Если бит j установлен, то определяемая штриховка может использоваться в ArchiCAD согласно ее определенному типу. По умолчанию для всей штриховки устанавливается (0).

pattern definition: pattern1, pattern2, pattern3, pattern4, pattern5, pattern6, pattern7, pattern8: 8 чисел в интервале 0 - 255, представляющие двоичные значения. Определяет растровый образец штриховки.



**spacing:** плотность штриховки - определяет глобальный масштабный множитель для всей штриховки. Все значения в обоих направлениях, как по оси  $x$ , так и по оси  $y$ , будут умножаться на величину, задаваемую этим коэффициентом.

**angle:** глобальный угол поворота в градусах.

**n:** количество линий штриховки.

**frequency $_i$ :** частота линии (расстояние между двумя линиями равно  $\text{spacing} * \text{frequency}_i$ ).

**dir $_i$ :** угол наклона линии в градусах.

**offset $_x_i$ , offset $_y_i$ :** смещение линии относительно начала координат.

**m $_i$ :** количество частей линии.

**length $_{ij}$ :** длина частей линии (фактическая длина равна  $\text{spacing} * \text{length}_{ij}$ ). Части линии - это следующие друг за другом отрезки и пробелы. Первой частью линии является отрезок, нулевая длина означает точку.

**Растровый рисунок** штриховки определяется параметрами `pattern1... pattern8` используется в ArchiCAD только, если в диалоге команды *Параметры/Вывод на экран* для параметра *Штриховка поверхностей* установлено значение *Растровый рисунок*. Для определения растрового рисунка выберите его элементарную единицу штриховки. Такая элементарная единица представляется точками и пробелами в прямоугольной сетке с 8 делениями по горизонтали и вертикали (8x8). Эти 8 параметров являются десятичными представлениями двоичных значений, определяющих рисунок штриховки (1 определяет точку, 0 - пробел).

**Векторная штриховка** является второй составляющей определения штриховки, Она задается в виде совокупности штриховых линий, повторяющихся с заданной частотой (`frequency $_i$` ). Каждая такая линия описывается направлением (`direction $_i$` ), смещением относительно начала координат (`offset $_x_i$ , offset $_y_i$` ) и определением собственно штриховой линии в виде последовательности чередующихся отрезков и пробелов заданной длины (`length $_{ij}$` ).

**Примечание:** В GDL с помощью `DEFINE FILL` определяется только простая штриховка. С помощью этой команды нельзя определить символов штриховки.

*Пример:*

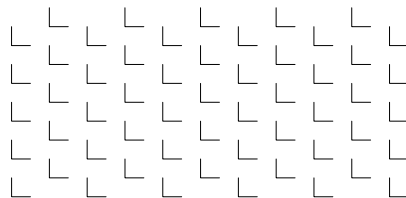
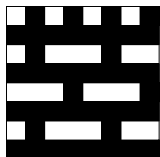
```
DEFINE FILL "brick" 85, 255, 136, 255,
  34, 255, 136, 255,
  0.08333, 0.0, 4,
  1.0, 0.0, 0.0, 0.0, 0,
  3.0, 90.0, 0.0, 0.0, 2,
  1.0, 1.0,
  3.0, 90.0, 1.5, 1.0, 4,
  1.0, 3.0, 1.0, 1.0,
  1.5, 90.0, 0.75, 3.0, 2,
  1.0, 5.0
```

*Растровый рисунок:*

Определение образца:	Двоичное представление:
pattern1 = 85	01010101     • • • •
pattern2 = 255	11111111     ••••••••
pattern3 = 136	10001000     •     •
pattern4 = 255	11111111     ••••••••
pattern5 = 34	00100010     •     •
pattern6 = 255	11111111     ••••••••
pattern7 = 136	10001000     •     •
pattern8 = 255	11111111     ••••••••

*Вид:*

*Векторная штриховка:*



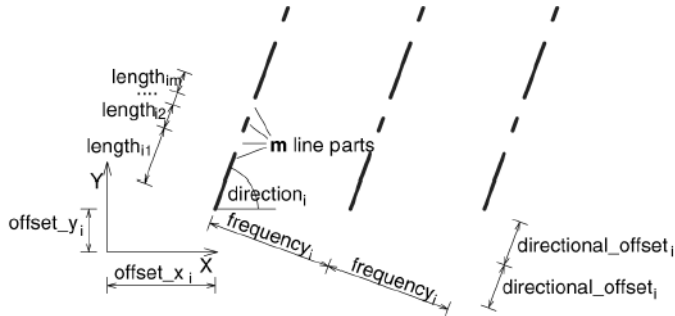
## DEFINE FILLA

```
DEFINE FILLA name [,] [FILLTYPES_MASK fill_types,] pattern1, pattern2, pattern3, pattern4,
  pattern5, pattern6, pattern7, pattern8, spacing_x, spacing_y, angle, n, frequency1,
  directional_offset1, direction1,
  offset_x1, offset_y1, m1, length11,
  ...
  length1m, ... frequencyn,
```

directional\_offsetn, directionn,  
 offset\_xn, offset\_yn, mn,  
 lengthn1, ... lengthnm

**Примечание:** Эта команда может содержать определение дополнительных данных.

См. *“Дополнительные данные”* на стр. 179 для получения дополнительной информации.

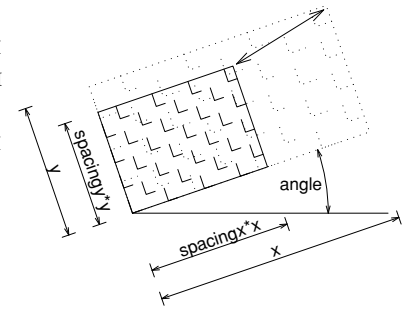


Расширение предложения DEFINE FILL.

*Дополнительные параметры*

spacing\_x, spacing\_y: коэффициент плотности штриховки соответственно по осям x и y. Эти два параметра определяют глобальный масштабный множитель для всей штриховки. Все значения по оси x будут умножаться на величину коэффициента spacing\_x. Точно также, все значения по оси y будут умножаться на величину коэффициента spacing\_y.

directional\_offseti: смещение начальной точки следующего аналогичного сегмента штриховки в направлении линии штриховки. Каждая линия сегмента штриховки вычерчивается на расстоянии, заданном коэффициентом frequency\_i, с учетом смещения, определяемого параметром directional\_offseti. Фактическая величина смещения определяется произведением: spacing \* directional\_offseti.



*Пример:*

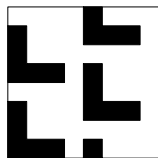
```
DEFINE FILLA "TEST" 8, 142, 128, 232,
  8, 142, 128, 232,
  0.5, 0.5, 0, 2,
  2, 1, 90, 0,
  0, 2, 1, 1,
  1, 2, 0, 0, 0,
  2, 1, 3
```

```
FILL "TEST"
POLY2 4, 6,
      -0.5, -0.5, 12, -0.5,
      12, 6, -0.5, 6
```

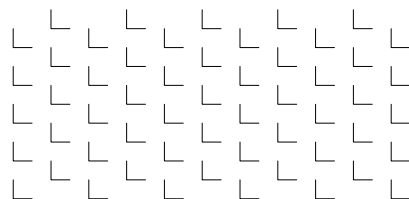
*Растровый рисунок:*

Определение образца:	Двоичное представление:
pat1 = 8	00001000     •
pat2 = 142	10001110     •   •••
pat3 = 128	10000000     •
pat4 = 232	11101000     ••• •
pat5 = 8	00001000     •
pat6 = 142	10001110     •   •••
pat7 = 128	10000000     •
pat8 = 232	11101000     ••• •

*Вид:*



*Векторная штриховка:*

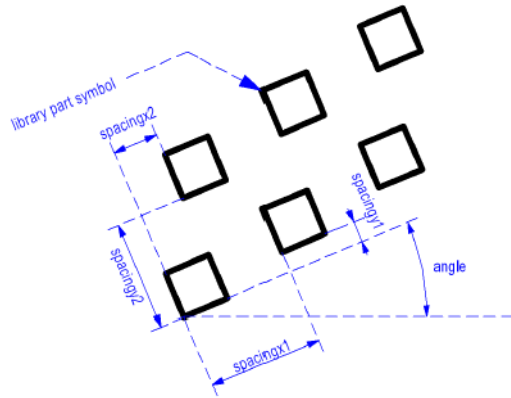


## DEFINE SYMBOL\_FILL

```
DEFINE SYMBOL_FILL name [,][FILLTYPES_MASK fill_types,]
  pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8,
  spacingx1, spacingy1, spacingx2, spacingy2,
  angle, scaling1, scaling2, macro_name [,] PARAMETERS [name1
  = value1, ... namen = valuen]
```

**Примечание:** Эта команда может содержать определение дополнительных данных.

См. *“Дополнительные данные”* на стр. 179 для получения дополнительной информации.



Является расширением предложения `DEFINE FILL`, которое позволяет включить чертеж библиотечного элемента в определение штриховки. Использование `masco_name` и параметров точно такое же, как и в команде `CALL`.

*Параметры:*

`spacingx1`, `spacingx2`: пробелы по горизонтали;

`spacingy1`, `spacingy2`: пробелы по вертикали;

`scaling1`: масштаб по горизонтали;

`scaling2`: масштаб по вертикали;

`masco_name`: имя библиотечного элемента.

## DEFINE SOLID\_FILL

```
DEFINE SOLID_FILL name [[,] FILLTYPES_MASK fill_types]]
```

Определение сплошной штриховки.

**Примечание:** Эта команда может содержать определение дополнительных данных.

См. *“Дополнительные данные”* на стр. 179 для получения дополнительной информации.

## DEFINE EMPTY\_FILL

```
DEFINE EMPTY_FILL name [[,] FILLTYPES_MASK fill_types]
```

Определение пустой штриховки.

**Примечание:** Эта команда может содержать определение дополнительных данных.

См. *“Дополнительные данные”* на стр. 179 для получения дополнительной информации.

## Определение типа линии

### DEFINE LINE\_TYPE

```
DEFINE LINE_TYPE name spacing, n,  
length1, ... lengthn
```

**Примечание:** Эта команда может содержать определение дополнительных данных.

См. *“Дополнительные данные” на стр. 179* для получения дополнительной информации.

Любой скрипт GDL может содержать определения типов линий до первого использования их названий. Тип линии, определенный таким образом, может использоваться только для 2D-элементов в самом скрипте и в скриптах нижнего уровня.

name: название типа линии.

spacing: коэффициент плотности.

n: количество частей линии.

lengthi: длина частей линии (фактическая длина равна  $\text{spacing} * \text{lengthi}$ ). Части линии - это следующие друг за другом отрезки и пробелы. Первой частью линии является отрезок, нулевая длина означает точку.

**Примечание:** В предложении DEFINE LINE\_TYPE можно определить только простой тип линии, т.е. линии, состоящие только из отрезков и пробелов. С помощью этой команды нельзя определить символьные типы линий.

*Пример:*

```
DEFINE LINE_TYPE "линия - - ." 1,  
6, 0.005, 0.002, 0.001, 0.002, 0.0, 0.002
```

### DEFINE SYMBOL\_LINE

```
DEFINE SYMBOL_LINE name dash, gap, macro_name PARAMETERS [name1 = value1, ... namen = valuen]
```

**Примечание:** Эта команда может содержать определение дополнительных данных.

См. *“Дополнительные данные” на стр. 179* для получения дополнительной информации.

Расширение предложения DEFINE LINE, которое предоставляет возможность включить чертеж библиотечного элемента в определение линии. Использование macro\_name и параметров точно такое же, как и в команде CALL.

*Параметры:*

dash: масштаб обоих компонент линии.

gap: зазор между каждой компонентой.

## Определение стиля

### DEFINE STYLE

**DEFINE STYLE** name font\_family, size, anchor, face\_code

Рекомендуется использовать с командами TEXT2 и TEXT.

Любой скрипт GDL может содержать определения стилей до первого использования их названий. Стиль, определенный таким образом, может использоваться только в самом скрипте и в скриптах нижнего уровня.

name: название стиля.

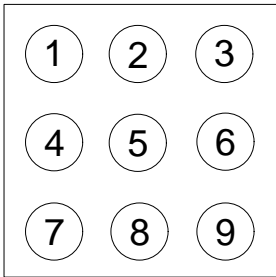
font\_family: название шрифтовой группы (например, Garamond.).

size: высота символа "I" в миллиметрах или в метрах в модельном пространстве.

Если определенный стиль используется с командами TEXT2 и TEXT, то size означает высоту символов в миллиметрах.

Если же стиль используется со строками PARAGRAPH в командах RICHTEXT2 и RICHTEXT, то size означает миллиметры или метры в зависимости от параметра fixed\_height предложения TEXTBLOCK, при этом значения "контурный" и "оттененный" в параметре face\_code и параметр anchor становятся недействительными.

anchor: код точки позиционирования текста.



face\_code: комбинация следующих значений:

- 0 обычный
- 1 **жирный**
- 2 *курсив*
- 4 подчеркнутый
- 8 контурный
- 16 оттененный

**Примечание:** Значения "контурный" и "оттененный" действуют только на платформе Macintosh и только в версиях ArchiCAD вплоть до 8.1.



## DEFINE STYLE {2}

**DEFINE STYLE{2}** name font\_family, size, face\_code

Новый вариант определения стиля, который рекомендуется использовать с определениями PARAGRAPH.

name: название стиля.

font\_family: название шрифтовой группы (например, Garamond.).

size: высота символов в миллиметрах или метрах в модельном пространстве.

face\_code: комбинация следующих значений:

0	обычный
1	<b>жирный</b>
2	<i>курсив</i>
4	<u>подчеркнутый</u>
32	надстрочный
64	подстрочный
128	<del>зачеркнутый</del>

Если определенный стиль используется с командой TEXT2, то параметр size означает высоту символов в миллиметрах, при этом значения "надстрочный", "подстрочный" и "зачеркнутый" параметра face\_code становятся недействительными. Если же стиль используется со строками PARAGRAPH в командах RICHTEXT2 и RICHTEXT, то size означает миллиметры или метры в зависимости от параметра fixed\_height предложения TEXTBLOCK.

## Определение абзаца

### PARAGRAPH

```
PARAGRAPH name alignment, firstline_indent,
left_indent, right_indent, line_spacing [,
tab_size1, ...]
[PEN index]
[[SET] STYLE style1]
[[SET] MATERIAL index]
'string1'
'string2'
...
'string n'
[PEN index]
[[SET] STYLE style2]
[[SET] MATERIAL index]
```

```
'string1'
'string2'
...
'string n'
...
```

**ENDPARAGRAPH**

Скрипты GDL могут включать определения абзацев до первого использования их названий. Абзац, определенный таким образом, может использоваться только в самом скрипте и в скриптах нижнего уровня. Абзац определяется как последовательность любого количества строк (каждая из которых не может превышать 256 символов) с различными реквизитами: стиль, перо и покрытие (в 3D). Если внутри абзаца не определяются реквизиты, то используются текущие (или по умолчанию) реквизиты. Символ новой строки (задаваемый с помощью набора символов '\n'), включенный в строку параграфа, приводит к автоматическому разбиению этой строки на два идентичных абзаца, каждый из которых содержит по одной строке. К определению абзаца можно сослаться по его имени в команде TEXTBLOCK. Все параметры абзаца, имеющие тип линейного размера (firstline\_indent, left\_indent, right\_indent, tab\_position), измеряются в миллиметрах или метрах в зависимости от параметра fixed\_height команды TEXTBLOCK. name: название абзаца.

alignment: выравнивание строк текста абзаца. Допустимые значения:

1: выравнивание влево, 2: выравнивание по центру, 3: выравнивание вправо, 4: выравнивание по ширине.

firstline\_indent: отступ первой строки в миллиметрах или метрах в модельном пространстве.

left\_indent: отступ слева в миллиметрах или метрах в модельном пространстве.

right\_indent: отступ справа в миллиметрах или метрах в модельном пространстве.

line\_spacing: коэффициент междустрочного интервала. По умолчанию расстояние между строками (размер символа + расстояние до следующей строки) определяется произведением фактического размера символов, определяемого стилем, на этот коэффициент.

tab\_positioni: последовательные позиции табуляции (каждая из которых указывается относительно начала абзаца), в миллиметрах или метрах в модельном пространстве. Нажатие клавиши табуляции приводит к перемещению курсора в текущую позицию табуляции. Если позиции табуляции не указываются, то используются значения по умолчанию (12,7 мм).

**Определение текстового блока****TEXTBLOCK**

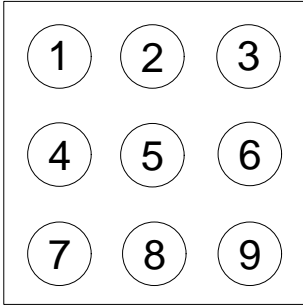
```
TEXTBLOCK name width, anchor, angle, width_factor, charspace_factor, fixed_height,
'string_expr1' [, 'string_expr2', ...]
```

Определение текстового блока. Любой скрипт GDL может содержать определения текстовых блоков до первого использования их

названий. Текстовый блок, определенный таким образом, может использоваться только в самом скрипте и в скриптах нижнего уровня. Текстовый блок - это последовательность произвольного количества строк или абзацев, которые могут быть размещены с использованием команд *“RICHTEXT2”* на стр. 128 и *“RICHTEXT”* на стр. 95. Используйте *“REQUEST (‘TEXTBLOCK\_INFO’, textblock\_name, width, height)”* для получения информации о вычисляемых значениях ширины и высоты TEXTBLOCK.

width: ширина текстового блока в миллиметрах или метрах в модельном пространстве, если указано 0, то ширина вычисляется автоматически.

anchor: код точки позиционирования текста.



angle: угол поворота текстового блока в градусах.

width\_factor: коэффициент ширины символов. Ширина символов, определяемая действующим стилем, умножается на этот коэффициент.

charspace\_factor: коэффициент расстояния между символами. Горизонтальное расстояние между соседними символами умножается на этот коэффициент.

fixed\_height: возможные значения:

1: размещенный TEXTBLOCK является независимым от масштаба и все указанные значения параметров линейного типа измеряются в миллиметрах.

0: размещенный TEXTBLOCK является зависимым от масштаба и все указанные значения параметров линейного типа измеряются в метрах в модельном пространстве.

string\_expr1: указывает имя ранее определенного абзаца, либо обычную строку в противном случае (которая принимает значения по умолчанию параметров абзаца).

## Дополнительные данные

Определения реквизитов могут содержать определения дополнительных данных, которые располагаются за ключевым словом ADDITIONAL\_DATA. Дополнительные данные должны определяться после других ранее определенных параметров команды

реквизита. Дополнительные данные имеют имя (namei) и значение (valuei), которые могут быть выражениями любого типа, и даже массивами. Если имя строкового параметра заканчивается подстрокой “\_file”, то значением этой строки является указанный файл, который включается в архивный проект. Различный смысл дополнительных данных может быть определен и использован в ArchiCAD или в расширениях ArchiCAD.

См. смысл параметров расширения LightWorks в <http://www.graphisoft.com/support/developer/documentation/LibraryDevDoc/10>.

Определение дополнительных данных имеется в следующих командах:

## **DEFINE MATERIAL**

```
DEFINE MATERIAL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
```

## **DEFINE FILL**

```
DEFINE FILL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
```

## **DEFINE FILLA**

```
DEFINE FILLA parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
```

## **DEFINE SYMBOL\_FILL**

```
DEFINE SYMBOL_FILL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
```

## **DEFINE LINE\_TYPE**

```
DEFINE LINE_TYPE parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
```

## **DEFINE SYMBOL\_LINE**

```
DEFINE SYMBOL_LINE parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
```

## **Зависимость от внешнего файла**

```
FILE_DEPENDENCE "name1" [, "name2", ...]
```

Вы можете предоставить список внешних файлов, от которых зависит Вам GDL-скрипт. Имя файла должно быть строковой константой.

Все указанные в таком списке файлы будут включены в архивный проект (точно так же, как и имена-константы макросов, используемые в предложениях CALL, и имена-константы рисунков, используемых в различных командах GDL). Эта команда действует только на уровне того скрипта, в котором она определена: если указанные файлы являются библиотечными элементами, то файлы из макровывозов не будут включены.

Эта команда может оказаться полезной в том случае, когда производится обращение к внешним файлам в различных местах скрипта GDL, например, в параметрах предложения ADDITIONAL\_DATA, в операциях работы с файлами.

---

# НЕГЕОМЕТРИЧЕСКИЕ СКРИПТЫ

*Помимо 3D- и 2D-скриптов, которые определяют внешний вид объектов GDL, имеются другие скрипты, которые добавляют к объекту дополнительную информацию. К ним относятся скрипты спецификаций, используемые для проведения количественных вычислений, скрипты параметров, которые содержат списки возможных значений различных параметров, и скрипты интерфейса пользователя, создающие дружелюбный интерфейс для ввода параметров. В этой главе описываются команды скриптов этих типов.*

## СКРИПТ СПЕЦИФИКАЦИЙ

Библиотечные элементы имеют специальное окно для скриптов спецификаций. Эти скрипты позволяют создавать зависимые от параметров библиотечные элементы спецификаций и, с помощью специальных директив, определять их размещение в смете. Используя в скрипте всего несколько команд, можно задать локальные компоненты и дескрипторы, характеристики, аналогичные тем, которые определялись в окне спецификаций предыдущих версий ArchiCAD. Вы можете также ссылаться на дескрипторы и компоненты из внешних баз данных. Длина поля кода не может превышать 32 символов. Кроме того, в скрипте спецификаций Вы можете использовать любые команды GDL, не предназначенные для построения фигур.

## DATABASE\_SET

**DATABASE\_SET** set\_name [descriptor\_name, component\_name, unit\_name, key\_name, criteria\_name, list\_set\_name]

Определение набора базы данных или выбор набора базы данных. Если эта команда размещена в скрипте MASTER\_GDL, то она будет определять набор базы данных, содержащий файлы дескрипторов, компонент, единиц измерения, ключей, критериев и сметных заданий.

На имя этого набора базы данных можно затем сослаться из скриптов спецификаций с использованием такой же команды, содержащей только параметр set\_name в качестве директивы, путем выбора фактического набора базы данных, который ссылается на REF COMPONENT и REF DESCRIPTOR. Имя набора базы данных по умолчанию - "Default Set", и оно будет использоваться, если никакой другой набор не выбран. Именами файлов по умолчанию для набора базы данных являются следующие: DESCDATA, COMPDATA, COMPUNIT, LISTKEY, LISTCRIT, LISTSET.

Скрипт может содержать выбор любого количества DATABASE\_SET.

set\_name: имя набора базы данных.

descriptor\_name: имя файла данных дескрипторов.

component\_name: имя файла данных компонент.

unit\_name: имя файла данных единиц измерения.

key\_name: имя файла данных ключей.

criteria\_name: имя файла критериев.

list\_set\_name: имя файла сметных заданий.

## DESCRIPTOR

**DESCRIPTOR** name [,code, keycode]

Определение локального дескриптора. Скрипт может включать произвольное число команд DESCRIPTOR.

name: может располагаться более чем на одной строке. Переход на новую строку определяется символом '\n' и '\п' а табуляция - символом '\t'. Вставка символов '\ ' в конце строки позволяет продолжать символьную константу на следующей строке, без добавления символа новой строки. Пара символов '\\ ' просто означает наличие в строке символа '\ '.

Длина строки (включая символы перехода на новую строку) не должна превышать 255 символов; все лишние символы будут отсечены компилятором. Если Вам необходимо задать текст большей длины, используйте несколько команд DESCRIPTOR.

code: строка, определяющая код дескриптора.

keycode: строка, осуществляющая ссылку на ключ во внешней базе данных.

Ключ назначается дескриптору.

## REF DESCRIPTOR

**REF DESCRIPTOR** code [, keycode]

Обращение к дескриптору из внешней базы данных по коду и ключевому коду.

## COMPONENT

**COMPONENT** name, quantity, unit [, proportional\_with, code, keycode, unitcode]

Описание локальной компоненты. Скрипты могут содержать любое количество команд COMPONENT.

name: название компоненты (не более 128 символов).

quantity: количество компоненты, числовое выражение.

unit: строка, используемая для определения единицы измерения.

proportional\_with: код от 1 до 6. При составлении сметы определенное выше количество компоненты будет автоматически умножаться на значение величины той составляющей элемента, которая определяется указанным кодом. Другими словами, с помощью данного кода Вы указываете в расчетах на какую характеристику элемента задается количество компоненты в quantity: Смысл кодов следующий:

- 1: элемент,
- 2: длина,

- 3: поверхность A,
- 4: поверхность B,
- 5: площадь поверхности,
- 6: площадь.

code: строка, определяющая код компоненты.

keycode: строка, осуществляющая ссылку на ключ во внешней базе данных. Ключ назначается компоненте.

unitcode: строка, ссылающаяся на единицу измерения во внешней базе данных. С помощью этого параметра контролируется выходной формат представления количества компоненты. Этот параметр замещает значение локально определенной единицы измерения.

## REF COMPONENT

**REF COMPONENT** code [, keycode [, numeric\_expression]]

Обращение к компоненте из внешней базы данных по коду и ключевому коду. Если задан параметр numeric\_expression, то при составлении сметы определенное в компоненте ее количество quantity будет умножаться не на величину согласно proportional\_with, а на числовое выражение, заданное в numeric\_expression.

## BINARYPROP

### **BINARYPROP**

Binaryprop - это ссылка на двоичные данные спецификаций (компоненты и дескрипторы), определенные в библиотечном элементе в разделах Компоненты и Дескрипторы.

Директивы DATABASE\_SET не оказывают влияния на двоичные данные.

## SURFACE3D ( )

**SURFACE3D** ( )

Функция Surface3D() определяет площадь поверхности 3D-фигуры библиотечного элемента.

**Внимание:** Если Вы разместили две или более фигуры в одном и том же месте с одинаковыми параметрами, то эта функция будет определять общую суммарную площадь поверхности.

## VOLUME3D ( )

**VOLUME3D** ( )

Функция Volume3D() определяет объем 3D-фигуры библиотечного элемента.

**Внимание:** Если Вы разместили две или более фигуры в одном и том же месте с одинаковыми параметрами, то эта функция будет определять общий суммарный объем.

## POSITION

**POSITION** position\_keyword

Эта команда действует только в смете компонент.

Изменяет только тип элемента, с которым будут ассоциироваться последующие дескрипторы и компоненты. Если такие директивы не заданы в скрипте спецификаций, то дескрипторы и компоненты будут перечисляться в соответствии с их типом, установленным по умолчанию.

*Команда имеет следующие значения параметра:*

WALLS  
COLUMNS  
BEAMS  
DOORS  
WINDOWS  
OBJECTS  
CEILS  
PITCHED\_ROOFS  
LIGHTS  
HATCHES  
ROOMS  
MESHES

Директива остается в силе для всех последующих команд DESCRIPTOR и COMPONENT до тех пор, пока не встретится следующая директива. Скрипт может включать любое количество таких директив.

*Пример:*

```
DESCRIPTOR "\tНавесной шкаф.\n\t Спецификации:\n\t\t\t - открывающиеся двери\n\t\t\t - регулируемая высота\n\t\t\t - водонепроницаемое покрытие"  
REF DESCRIPTOR "0001"  
s = SURFACE3D () ! площадь поверхности  
COMPONENT "клей", 1.5, "kg"  
COMPONENT "ручка", 2*c, "nb"! c - количество дверей  
COMPONENT "краска", 0.5 * s, "кг"  
POSITION WALLS  
REF COMPONENT "0002"
```

## DRAWING

**DRAWING**

DRAWING: -ссылка на чертеж, описанный в 2D-скрипте того же библиотечного элемента. Используется для помещения рисунков чертежей в смету.



## СКРИПТ ПАРАМЕТРОВ

Список значений параметра - это множество возможных числовых или строковых значений. Такой список может быть приписан параметрам, определенным в скрипте параметров библиотечного элемента или в скрипте MASTER\_GDL. Тип параметра должен быть списком значений любого простого типа. Совместимость типов проверяется компилятором GDL.

Всякий раз при изменении значения какого-либо параметра, принимающего значение из списка, производится интерпретация скрипта списка значений и определенный в скрипте перечень возможных значений будет выводиться во всплывающем меню.

### VALUES

```
VALUES "fillparam_name" [[,] FILLTYPES_MASK fill_types,] [,]value_definition1
    [, value_definition2, ...]
```

name: имя параметра,

fill\_types = j1 + 2 \* j2 + 4 \* j3

j1: штриховка сечений.

j2: штриховка поверхностей.

j3: штриховка чертежей.

Может использоваться для параметров типа штриховки. Если бит j установлен, то всплывающее меню штриховки, соответствующее параметру "fillparam\_name", будет автоматически содержать только штриховку указанного типа. По умолчанию используется вся штриховка (0).

value\_definitioni: определение значения; может быть:

expressioni: числовое или строковое выражение, или

**CUSTOM:** ключевое слово, означает, что может быть введено любое специальное значение, или

**RANGE** left\_delimiter [expression<sub>1</sub>], [expression<sub>2</sub>]  
 right\_delimiter [STEP step\_start\_value,  
 step\_value]

Определение диапазона с факультативным шагом

left\_delimiter: [, означает '>=', или (, означает '>'

expression1: выражение нижнего значения диапазона.

expression2: выражение верхнего значения диапазона

right\_delimiter: ], означает '<=', или ), означает '<'

step\_start\_value: начальное значение

step\_value: значение шага

*Примеры:*

```

VALUES "par1" 1, 2, 3
VALUES "par2" "a", "b"
VALUES "par3" 1, CUSTOM, SIN (30)
VALUES "par4" 4,RANGE(5, 10],12,RANGE(,20]
    STEP 14.5, 0.5, CUSTOM

! Пример чтения всех строковых значений из файла
! и использование его в списке значений
DIM sarray[]
! файл в библиотеке, который содержит данные параметра
filename = "ProjectNotes.txt"
ch1 = OPEN ("text", filename, "MODE=RO, LIBRARY")
i = 1
j = 1
sarray[1] = ""
! сбор всех строк
DO
n = INPUT (ch1, i, 1, var)
IF n > 0 AND VARTYPE (var) = 2 THEN
sarray[j] = var
j = j + 1
ENDIF
i = i + 1
WHILE n > 0
CLOSE ch1
! список значений параметра со строками, прочитанными из файла
VALUES "RefNote" sarray

```

**PARAMETERS**

```

PARAMETERS name1 = expression1 [,
    name2 = expression2, ...,
    namen = expressionn]

```

namei: имя параметра.

expressioni: новое значение параметра

С помощью этой команды можно изменять значения параметра с использованием скрипта параметров.

Выполненные изменения окажутся действительными только при следующей интерпретации. Команды в макросе ссылаются на параметры того скрипта, который вызвал этот макрос. Если параметр является списком значений, то выбранное значение будет либо существующим значением, либо специальным значением, либо первым значением из списка значений.

Кроме того, глобальная переменная GLOB\_MODPAR\_NAME содержит имя последнего измененного пользователем параметра.

## LOCK

**LOCK** name1 [, name2, ..., namen]

Блокирует указанные параметры в диалоговом окне установки параметров. Заблокированный параметр представляется в диалоговом окне серым цветом и пользователь не может изменять его значение.

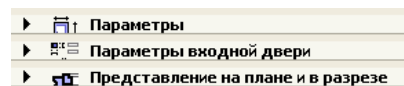
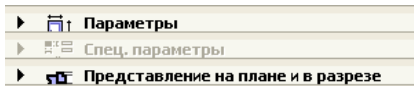
## HIDEPARAMETER

**HIDEPARAMETER** name1 [, name2, ..., namen]

Прячет указанные параметры и его дочерние параметры в диалоговом окне установки параметров. Параметр, спрятанный с помощью этой команды в скрипте параметров, автоматически исчезает из списка параметров.

## СКРИПТ ИНТЕРФЕЙСА ПОЛЬЗОВАТЕЛЯ

С помощью приводимых далее команд GDL можно определить специальный пользовательский интерфейс для панели *Специальные параметры* диалогового окна установки параметров библиотечного элемента. Если Вы нажмете кнопку *Установить по умолчанию* в редакторе библиотечного элемента, то по умолчанию будет использоваться специальный пользовательский интерфейс в диалоговом окне установки параметров объекта (двери, окна и т.д.). Параметры со специальным управлением не прячутся автоматически в исходном списке параметров, однако они могут быть спрятаны вручную в редакторе библиотечного элемента



Начало системы координат располагается в верхнем левом углу. Размеры и значения координат измеряются в пикселах.

## UI\_DIALOG

**UI\_DIALOG** title [, size\_x, size\_y]

Определяет заголовок диалогового окна. В настоящий момент доступная область фиксирована размерами 444 x 266 пикселей, поэтому параметры size\_x и size\_y не используются.

**Ограничение:** Скрипт интерфейса может содержать только одну команду UI\_DIALOG.

## UI\_PAGE

**UI\_PAGE** page\_number

Директива страницы; определяет страницу, на которой будут располагаться элементы интерфейса. Нумерация страниц начинается с 1. Перемещение между страницами может быть определена двумя способами. Первый способ заключается в использовании

кнопок, создаваемых с помощью команд `UI_NEXT` и `UI_PREV`. Второй вариант - создание динамического манипулирования страницами с помощью команды `UI_CURRENT_PAGE`.

Если в скрипте интерфейса нет команды `UI_PAGE`, то по умолчанию все элементы располагаются на первой странице.

**Внимание:** Любой разрыв непрерывности определения страниц приводит к вставке новой страницы без кнопок, поэтому нет возможности перейти из этой страницы на другую.

## UI\_CURRENT\_PAGE

### UI\_CURRENT\_PAGE index

Определение текущей страницы, выводимой на экран.

**Внимание:** Переход на несуществующую страницу приводит к вставке новой страницы без кнопок и управляющих элементов, поэтому нет возможности перейти из этой страницы на другую.

index: правильный индекс `UI_PAGE`, которую следует вывести на экран.

## UI\_BUTTON

`UI_BUTTON type, text, x, y, width, height [, id [, url]]`

Определение кнопки на текущей странице. Кнопки могут использоваться для различных целей: переход от страницы к странице, открытие страницы веб или выполнение некоторых действий.

type: тип кнопки. Имеются следующие типы:

`UI_PREV`: если кнопка нажата, то происходит переход на предыдущую страницу;

`UI_NEXT`: если кнопка нажата, то происходит переход на следующую страницу;

`UI_FUNCTION`: если кнопка нажата, то глобальная переменная `GLOB_UI_BUTTON_ID` принимает значение ID кнопки, определяемой выражением;

`UI_LINK`: если кнопка нажата, то открывается браузер с URL, определяемым выражением;

text: текст, располагающийся на кнопке;

x, y: расположение кнопки;

width, height: ширина и высота кнопки в пикселах;

id: уникальный идентификатор целого типа;

url: строка, содержащая URL.

Кнопки становятся недоступными, если предыдущая/следующая страница отсутствует. Если эти кнопки нажимаются, то параметр `gs_ui_current_page` библиотечного элемента принимает значение индекса страницы, которую следует показать - если имеется параметр с таким именем.

*Пример:*

```
! Скрипт UI
UI_CURRENT_PAGE gs_ui_current_page
UI_BUTTON UI_FUNCTION, "Перейти на стр. 9", 200,150, 70,20, 3
UI_BUTTON UI_LINK, "Посетить веб-сайт", 200,180, 100,20, 0,
http://www.opendesign.com/
! Скрипт параметров
if GLOB_UI_BUTTON_ID = 3 then
parameters gs_ui_current_page = 9, ...
endif
```

## UI\_SEPARATOR

**UI\_SEPARATOR** x1, y1, x2, y2

Создает разделяющий прямоугольник. Прямоугольник вырождается в одну (вертикальную или горизонтальную) разделительную линию, если  $x1 = x2$  или  $y1 = y2$

x1, y1: координаты верхней левой вершины (координаты начальной вершины прямоугольника).

x2, y2: координаты нижней правой вершины (координаты конечной вершины прямоугольника).

## UI\_GROUPBOX

**UI\_GROUPBOX** text, x, y, width, height

GROUPBOX - прямоугольник группирования. Может использоваться для визуального группирования логически взаимосвязанных параметров.

text: заголовок прямоугольника группирования.

x, y: координаты верхней левой вершины.

width, height: ширина и высота в пикселах.

## UI\_PICT

**UI\_PICT** expression, x, y [,width, height[, mask]]

Элемент-рисунок в диалоговом окне. Файл рисунка должен располагаться в одной из загруженных библиотек.

expression: имя файла или номер индекса рисунка, запоминаемого в библиотечном элементе. Индекс 0 относится к

рисунку предварительного просмотра библиотечного элемента.

x, y: место расположения верхнего левого угла рисунка.

width, height: факультативная ширина и высота рисунка; по умолчанию используются исходная ширина и высота рисунка.

mask = альфа + искажение.

*См. описание элемента "PICTURE" для ознакомления с подробным описанием.*

## UI\_STYLE

**UI\_STYLE** fontsize, face\_code

Все генерируемые вслед за этой командой предложения UI\_OUTFIELD и UI\_INFIELD будут представлять этот стиль до появления следующей команды UI\_STYLE.

fontsize: одно из следующих значений размера шрифта:

- 0: небольшой;
- 1: совсем маленький,
- 2: большой

face\_code: аналогичен такому же параметру в STYLE, однако нельзя использовать комбинацию значений.

- 0: обычный,
- 1: **жирный**,
- 2: *курсив*,
- 4: подчеркнутый,
- 8: контурный,
- 16: оттененный.

## UI\_OUTFIELD

**UI\_OUTFIELD** expression, x, y, width, height [, flags]]

Создается статический текст.

expression: выражение числового или строкового типа.

x, y: расположение верхнего левого угла текстового блока.

width, height: ширина и высота в пикселах.

flags = j1 + 2\*j2 + 4\*j3.

j1 (1) и j2 (2) - выравнивание по горизонтали.  
 j1 = 0, j2 = 0 : выравнивание по левому краю (по умолчанию).  
 j1 = 1, j2 = 0 : выравнивание по правому краю.  
 j1 = 0, j2 = 1 : выравнивание по центру.  
 j1 = 1, j2 = 1 : не используется.

j3 (4): - серый текст.

## UI\_INFIELD

```

UI_INFIELD "name", x, y, width, height [,
method, picture_name,
images_number,
rows_number, cell_x, cell_y,
image_x, image_y,
expression_image1, text1,
... ,
expression_imagen, textn]
  
```

## UI\_INFIELD {2}

```

UI_INFIELD{2} name, x, y, width, height [,
method, picture_name,
images_number,
rows_number, cell_x, cell_y,
image_x, image_y,
expression_image1, text1,
... ,
expression_imagen, textn]
  
```

## UI\_INFIELD {3}

```

UI_INFIELD{3} name, x, y, width, height [,
method, picture_name,
images_number,
rows_number, cell_x, cell_y,
image_x, image_y,
expression_image1, text1, value_definition1,
... ,
expression_imagen, textn, value_definitionn]
  
```

Создается редактируемый текст или всплывающее меню для выбора параметров. Всплывающее меню создается в том случае, если параметр имеет тип списка значений, покрытия, штриховки, типа линии или цвета пера.

Если в команде присутствуют факультативные параметры, то список значений имеет альтернативное представление в виде пиктограмм. Имеются различные пиктографические управляющие элементы. Они представляются указанными пиктограммами и текстами и позволяют одновременно выбрать не более одного элемента, точно так же, как и во всплывающем меню.

В вариантах 1 и 2 этой команды пиктографические элементы и элементы списка значений ассоциируются по индексам.

В варианте 3 эта команда определяет ассоциацию значений, которая связывает пиктографические элементы с элементами списка значений ассоциированного с ними параметра. Если значение, определенное в пиктографическом элементе, отсутствует в списке значений параметра, то оно не будет представлено в управляющем элементе.

После изменения любого из параметров скрипт интерфейса компонуется заново с новым значением.

**name:** имя параметра в виде строкового выражения для UI\_INFIELD или имя параметра с факультативным фактическим значением индекса, если используется массив для UI\_INFIELD{2}.

**x, y:** расположение редактируемого текста, всплывающего меню или управляющего элемента.

**width, height:** ширина и высота в пикселах.

**method:** тип пиктографического управляющего элемента:

- 1: список пиктографических изображений;
- 2: всплывающее меню;
- 3: альтернативная пиктографическая кнопка,
- 4: нажимаемая пиктографическая кнопка.

**picture\_name:** имя файла изображений, содержащего матрицу конкатенированных изображений или пустую строку.

**images\_number:** номер изображения в матрице.

**rows\_number:** количество строк в матрице.

**cell\_x, cell\_y:** ширина и высота ячейки внутри поля представления пиктограммы, включая само изображение и соответствующий ему текст.

**image\_x, image\_y:** ширина и высота изображения в ячейке.

**expression\_image*i*:** индекс изображения с номером *i* в матрице, или индивидуальное имя файла. Если специфицировано имя файла изображения, то здесь следует использовать индексы. Нельзя комбинировать индексы с индивидуальными именами файлов.

**text*i*:** текст в ячейке с номером *i*.

**value\_definition*i*:** определение значения, которое соответствует по значению ячейке с элементом в виде списка значений:

**expression:** числовое или строковое выражение или

**CUSTOM:** ключевое слово; означает, что может быть введено любое требуемое значение



## Пример 1:

```

IF с THEN
  UI_DIALOG "Параметры определения отверстия"
  UI_OUTFIELD "Тип отв-тия:", 15, 40, 180, 20
  UI_INFIELD "D", 190, 40, 105, 20
  IF D="прямоугольное" THEN
    UI_PICT "rect.pict", 110, 33, 60, 30
    UI_OUTFIELD "Ширина отверстия:", 15, 70, 180, 20
    UI_INFIELD "E", 190, 70, 105, 20
    UI_OUTFIELD "Высота отверстия:", 15, 100, 180, 20
    UI_INFIELD "F", 190, 100, 105, 20
    UI_OUTFIELD "Расст. между отв-тиями:", 15, 130, 180, 20
    UI_INFIELD "G", 190, 130, 105, 20
  ELSE
    UI_PICT "circle.pict", 110, 33, 60, 30
    UI_OUTFIELD "Диаметр кругл. отверстия:", 15, 70, 180, 20
    UI_INFIELD "J", 190, 70, 105, 20
    UI_OUTFIELD "Расст. м. ц. отверстий:", 15, 100, 180, 20
    UI_INFIELD "K", 190, 100, 105, 20
    UI_OUTFIELD "Аппрокс. кругл. отв-тия:", 15, 130, 180, 20
    UI_INFIELD "M", 190, 130, 105, 20
  ENDIF
  UI_OUTFIELD "Количество отверстий:", 15, 160, 180, 20
  UI_INFIELD "I", 190, 160, 105, 20
ENDIF
UI_SEPARATOR 50, 195, 250, 195
UI_OUTFIELD "Покрытие балки:", 15, 210, 180, 20
UI_INFIELD "MAT", 190, 210, 105, 20
UI_OUTFIELD "Перо балки:", 15, 240, 180, 20
UI_INFIELD "P", 190, 240, 105, 20

```

Параметры определения отверстия

Тип отв-тия: Прямоуголь

Ширина отверстия: 0.500

Высота отверстия: 0.500

Расст. между отв-тиями: 0.200

Количество отверстий: 3

Покрытие балки:  Plaster, s...

Перо балки:

Параметры определения отверстия

Тип отв-тия: Круглое

Диаметр кругл. отверстия: 0.500

Расст. м. ц. отверстий: 0.800

Аппрокс. кругл. отв-тия: 24

Количество отверстий: 3

Покрытие балки:  Plaster, s...

Перо балки:

*Пример 2:*

```
! Скрипт параметров:  
VALUES "myParameter" "Два", "Три", "Пять", CUSTOM
```

```
! Скрипт интерфейса:  
px = 80  
py = 60  
cx = px + 3  
cy = py + 25
```

```
UI_INFIELD{3} "myParameter", 10, 10, 4 * cx + 21, cy + 5,  
1, "myPicture", 6,  
1, cx, cy, px, py,  
1, "1 - один", "Один",  
2, "2 - два", "Два",  
3, "3 - три", "Три",  
4, "4 - четыре", "Четыре",  
5, "5 - пять", "Пять",  
6, "специальное значение", CUSTOM
```

## UI\_TOOLTIP

```
UI_BUTTON type, text, x, y, width, height [, id [, url]] [ UI_TOOLTIP tooltiptext ]  
UI_INFIELD "name", x, y, width, height [, extra parameters ... ] [ UI_TOOLTIP tooltiptext ]  
UI_INFIELD{2} name, x, y, width, height [, extra parameters ... ] [ UI_TOOLTIP tooltiptext ]  
UI_INFIELD{3} name, x, y, width, height [, extra parameters ... ] [ UI_TOOLTIP tooltiptext ]  
UI_OUTFIELD expression, x, y, width, height [, flags] [ UI_TOOLTIP tooltiptext ]  
UI_PICT expression, x, y [,width, height [, mask]] [ UI_TOOLTIP tooltiptext ]
```

Определяется подсказка управляющего элемента страницы интерфейса пользователя. Подсказки допустимы для кнопок, полей ввода (infield), полей вывода (outfield) и рисунков.

tooltiptext: текст, выводимый в качестве подсказки к управляющему элементу.

---

# ВЫРАЖЕНИЯ И ФУНКЦИИ

*Все параметры в фигурах GDL могут получаться в результате вычислений. Например, Вы можете указать, что высота цилиндра в пять раз больше его радиуса, или, прежде чем определить куб, Вы можете сместить систему координат в трех направлениях на величину, равную половине его стороны, чтобы начало координат оказалось в середине куба, а не в нижнем левом углу. Для задания таких вычислений в языке GDL существуют различные арифметические конструкции: выражения, операторы, функции.*

## ВЫРАЖЕНИЯ

В предложениях GDL можно использовать сложные выражения. Выражения могут быть числового или строкового типа. ни могут включать константы, переменные, параметры обращения к функциям, а также любую комбинацию выше перечисленных компонент, соединенных операторами. Пара круглых скобок (()) (приоритет 1) используется для изменения приоритета операторов.

Переменные **простых** типов могут быть заданы как числовыми, так и строковыми значениями, даже в рамках одного и того скрипта, и соответственно, их можно использовать в выражениях числового и строкового типа. Строковые выражения НЕЛЬЗЯ использовать непосредственно в качестве имен макросов в макровывогах и в качестве названий реквизитов при определении покрытий, штриховок, типов линий или стилей. Переменные, для которых заданы строковые значения, можно использовать во всех тех местах, где синтаксис языка допускает наличие строкового значения. Если в дальнейшем в скрипте этой переменной присвоено числовое значение, то ее уже можно использовать в числовых выражениях до тех пор, пока ее значение опять не будет переопределено на строку. Тип выражений проверяется в процессе предварительной компиляции.

GDL поддерживает одномерные и двумерные массивы. Переменные становятся массивами после выполнения предложения объявления, в котором определяются максимальные размеры массива.

## DIM

```
DIM var1[dim_1], var2[dim_1][dim_2], var3[ ],  
      var4[ ][ ], var5[dim_1][ ],  
      var5[ ][dim_2]
```

После ключевого слова DIM перечисляются через запятую переменные. var1, var2, ... - это имена массивов, а заключенные в квадратные скобки числа представляют размерности массива (числовые константы). Для определения размерности массива запрещено использовать выражения с переменными. Если они опущены, то массив объявляется динамическим (одно или оба измерения).

Параметры библиотечного элемента также могут быть массивами. Их размерность определяется в диалоге установки параметров библиотечного элемента. Параметры-массивы не должны объявляться в скрипте и они становятся динамическими по умолчанию. При вызове библиотечного элемента с помощью предложения CALL подставляемые фактические значения параметра-массива могут быть массивами произвольных размеров.

К элементам массивов можно обращаться в любом месте скрипта, однако, если они являются переменными, то только после их объявления.

```
var1[num_expr] или var1  
var2[num_expr1][num_expr2] или var2[num_expr1]  
или var2
```

Имя массива без указания значений индексов обозначает ссылку на весь массив (или на строку двумерного массива), что приемлемо в целом ряде случаев (предложения CALL, PRINT, LET, PUT, REQUEST, INPUT, OUTPUT, SPLIT). Для динамических массивов отсутствуют ограничения на фактическое значение индекса. В процессе интерпретации, когда несуществующему элементу динамического массива задается значение, происходит выделение соответствующего участка памяти и отсутствующим элементам устанавливается значение 0 (числовое).

**Внимание!** В некоторых случаях это может вызвать ошибку, связанную с отсутствием памяти. Любой индекс - даже возможно неправильный, имеющий огромное значение - рассматривается правильным, так как интерпретатор не в состоянии обнаружить ошибочное условие, связанное с такой ситуацией. Несуществующий элемент динамического массива равен 0 (числовое).

Массивы, имеющие фиксированный размер, проверяются на соответствие фактического индекса фиксированному размеру. Переменным массива фиксированного размера не можно присваивать значения динамического массива. Однако в динамических массивах это можно делать. Это правило также применяется к некоторым предложениям, в которых в качестве возвращаемого параметра указывается ссылка на весь массив. (REQUEST, INPUT, SPLIT).

Элементы массива могут использоваться в любом числовом или строковом выражении. Элементы массива могут быть заданы числовыми или строковыми значениями.

В качестве индекса может использоваться любое числовое выражение, индексация начинается с 1.

Элементы массива могут быть любого простого типа (числовой, строковый, группа). Тип всего массива ('основной' тип) - это тип его первого элемента ([1] или [1][1]). Массивы параметров и глобальных переменных не могут быть смешанного типа.

## VARDIM1(expr)

**VARDIM1** (expr)

## VARDIM2(expr)

**VARDIM2** (expr)

Эти функции возвращают целое значение, которое является значением фактического размера выражения типа массива, представленного параметром. Их следует использовать в том случае, если Вы хотите правильно оперировать элементами динамического массива или параметрами массива. Если динамическому массиву еще не было установлено ни одного элемента, то эти функции возвращают значение 0. Для одномерных массивов VARDIM2 возвращает значение 0.

*Примеры числовых выражений:*

```
Z
5.5
(+15)
-X
A*(B+C)
SIN(X+Y)*Z
A+R*COS(I*D)
5' 4"
SQR (x^2 + y^2) / (1 - d)
a + b * sin (alpha)
height * width
```

*Примеры строковых выражений:*

```
"Строковая константа"
name + STR ("%m", i) + "." + ext
string_param <> "Mode 1"
```

*Примеры выражений с массивами:*

```
DIM tab [5], tab2 [3][4] ! объявление
tab [1] + tab [2]
tab2 [2][3] + A
PRINT tab

DIM f1 [5], v1[], v2[][]
v1[3] = 3 !v1[1] = 0, v1[2] = 0, array of 3
! элементы
v2[2][3] = 23 ! все другие элементы (2 X 3) = 0
PRINT v1, v2

DIM f1 [5], v1[], v2[][]
FOR i = 1 TO VARDIM1(f1)
    f1 [i] = i
NEXT I
v1 = f1
v2 [1] = f1
PRINT v1, v2
```

## ОПЕРАТОРЫ

Ниже приводятся операторы в порядке уменьшения их приоритета. Вычисление выражения начинается с оператора с наивысшим приоритетом, а в случае нескольких подряд расположенных операторов одного приоритета вычисление производится слева направо.

### Арифметические операторы

<b>^</b> (или <b>**</b> )	сепень	приоритет 2
<b>*</b>	умножение	приоритет 3
<b>/</b>	деление	приоритет 3
<b>MOD</b> (или <b>%</b> )	остаток от деления $X \text{ MOD } Y = X - Y * \text{INT} (X/Y)$	приоритет 3
<b>+</b>	сложение	приоритет 4
<b>-</b>	вычитание	приоритет 4

**Примечание:** операция "+" применима и в выражениях строкового типа: результатом в данном случае является конкатенация строк.

Результатом оператора '/' (деление) всегда является действительное число. Для других операторов результат зависит от типа операндов: если все операнды целого типа, то и результат целого типа, в противном случае - действительного типа.

### Операторы отношения

<b>=</b>	равно	приоритет 5
<b>&lt;</b>	меньше	приоритет 5
<b>&gt;</b>	больше	приоритет 5
<b>&lt;=</b>	меньше или равно	приоритет 5
<b>&gt;=</b>	больше или равно	приоритет 5
<b>&lt;&gt;</b> (или <b>#</b> )	не равно	приоритет 5

**Примечание:** Отношения могут использоваться со строковыми выражениями. Они возвращают значения 0 или 1. Операторы действуют с учетом регистра буквенных символов (прописные и строчные). Не рекомендуется использовать операторы '=' (равно), '<=' (меньше или равно), '>=' (больше или равно), '<>' (или '#') (не равно) с операндами действительного типа, так как их выполнение может вызвать проблемы в связи с точностью представления таких чисел.

### Логические операторы

<b>AND</b> (или <b>&amp;</b> )	логическое И	приоритет 6
<b>OR</b> (или <b> </b> )	логическое включающее ИЛИ	приоритет 7
<b>EXOR</b> (или <b>@</b> )	логическое исключаящее ИЛИ	приоритет 8

**Примечание:** Логические операторы работают с целыми числами. Так 0 обозначает “ложь”, а любое другое число - “истина”. Значением логического выражения также является целое число, то есть, 1 обозначает “истина”, а 0 - “ложь”. Не рекомендуется использовать логические операторы с операндами действительного типа, так как в этом случае могут возникнуть проблемы в связи с точностью представления таких чисел.

## ФУНКЦИИ

### Арифметические функции

#### ABS

**ABS** (x) возвращает абсолютное значение x (целое значение, если x - целое число, и действительное число в противном случае).

#### CEIL

**CEIL** (x) возвращает наименьшее целое число, которое не меньше x (всегда целое число).  
Например,  $CEIL(1.23) = 2$ ;  $CEIL(-1.9) = -1$ .

#### INT

**INT** (x) возвращает целую часть x (всегда целое число). Например,  $INT(1.23) = 1$ ,  $INT(-1.23) = -2$ .

#### FRA

**FRA** (x) возвращает дробную часть x (целое число 0, если x - целое, и действительное число в противном случае). Например,  $FRA(1.23) = 0.23$ ,  $FRA(-1.23) = 0.77$ .

#### ROUND\_INT

**ROUND\_INT** (x) возвращает округленную целую часть числа x. Выражение 'i = ROUND\_INT (x)' эквивалентно следующему скрипту:  
`IF x < 0.0 THEN i = INT (x - 0.5) ELSE i = INT (x + 0.5)`

#### SGN

**SGN** (x) возвращает целое число +1, если x - положительное число, и целое число -1, если x - отрицательное число; в противном случае возвращается целое число 0.

#### SQR

**SQR** (x) возвращает корень квадратный x (всегда действительное число).

## Тригонометрические функции

Эти функции используют градусы в качестве аргументов (COS, SIN, TAN) и в качестве результата (ACS, ASN, ATN).

### ACS

**ACS** (x) возвращает арккосинус x. ( $-1.0 \leq x \leq 1.0$ ;  $0^\circ \leq \text{ACS}(x) \leq 180^\circ$ ).

### ASN

**ASN** (x) возвращает арксинус x. ( $-1.0 \leq x \leq 1.0$ ;  $-90^\circ \leq \text{ASN}(x) \leq 90^\circ$ ).

### ATN

**ATN** (x) возвращает арктангенс x. ( $-90^\circ \leq \text{ATN}(x) \leq 90^\circ$ ).

### COS

**COS** (x) возвращает косинус x.

### SIN

**SIN** (x) возвращает синус x.

### TAN

**TAN** (x) возвращает тангенс x.

### PI

**PI** возвращает константу Лудольфа. ( $p = 3.1415926\dots$ ).

**Примечание:** Все возвращаемые значения являются действительного типа.

## Трансцендентные функции

### EXP

**EXP** (x) возвращает e в степени x ( $e = 2.7182818$ ).

### LGT

**LGT** (x) возвращает логарифм x по основанию 10.



## LOG

**LOG** (x) возвращает натуральный логарифм x.

**Примечание:** Все возвращаемые значения являются действительного типа.

## Логические функции

### NOT

**NOT** (x) возвращает "ложь" (= 0 - целое число), если x "истина" ( $>0$ ), и "истина" (=1 - целое число), если x "ложь" (=0). (Логическое отрицание).

**Примечание:** Значение параметра должно быть целого типа.

## Статистические функции

### MIN

**MIN** (x1, x2, ... xn) возвращает наименьший из аргументов. Количество аргументов не ограничено.

### MAX

**MAX** (x1, x2, ... xn) возвращает наибольший из аргументов. Количество аргументов не ограничено.

### RND

**RND** (x) возвращает случайное значение действительного типа между 0.0 и x. ( $x > 0.0$ ).

## Битовые функции

### BITTEST

**BITTEST** (x, b)

Возвращает 1, если бит b числа x установлен в 1, и 0 в противном случае.

### BITSET

**BITSET** (x, b [, expr])

expr может принимать значение 0 или отличающееся от него; по умолчанию предполагается значение 1. Устанавливает бит b числа x в 1 или 0 в зависимости от значения выражения expr; возвращает число, полученное в результате такой замены. Параметр должен быть целого типа, возвращаемое значение - целого типа.

## Специальные функции

Специальные функции (как и глобальные переменные) могут использоваться в скрипте для обмена данными с ArchiCAD. Эти функции либо запрашивают текущее состояние и другие установки параметров рабочей среды ArchiCAD, либо обращаются к текущим параметрам библиотечного элемента. Запросы тоже можно использовать для обмена данными с GDL.

Имеется два типа специальных функций: запросы и функции IND.

**REQ** (parameter\_string)

**REQUEST** (question\_name, name | index, variable1 [, variable2,...])

**IND** (MATERIAL, name\_string)

**IND** (FILL, name\_string)

**IND** (LINE\_TYPE, name\_string)

**IND** (STYLE, name\_string)

**IND** (TEXTURE, name\_string)

Возвращаемое запросом значение - это всегда количество успешно найденных значений (целого типа), а тип найденных значений определяется каждым запросом отдельно.

Функции IND возвращают индекс реквизита (целого типа).

*Для получения дополнительной информации см. описание функции IND в “Разное” > “Запросы” на стр. 245.*

## Строковые функции

### STR

**STR** (numeric\_expression, length, fractions)

### STR

**STR** (format\_string, numeric\_expression)

### STR{2}

**STR{2}** (format\_string, numeric\_expression [, extra\_accuracy\_string])

Первая функция преобразует в символьную строку текущее значение числового выражения numeric\_expression. Параметр length - максимальное количество цифровых символов в строке, а fractions - количество дробных знаков. Если сконвертированное значение имеет больше символов, чем length, то оно усекается должным образом. Если же оно имеет меньше символов, то заполняется слева (length > 0) или справа (length < 0).

*Пример:*

```
A=4.5
B=2.345
ТЕХТ2 0, 2, STR(A, 8, 2)      ! 4.50
ТЕХТ2 0, 1, STR(B, 8, 2)      ! 2.34
ТЕХТ2 0, 0, STR(A*B, 8, 2)    ! 10.55
```

Во втором и третьем вариантах этой команды параметр `format_string`, предназначенный для указания характера форматирования строки, может быть либо переменной, либо константой. Если формат пустой, то числовое значение переводится в метры с точностью до трех десятичных знаков (с показом 0 целых).

Если в `format_string` установлены флажки дополнительной точности, то функция `STR{2}` будет возвращать соответствующую строку дополнительной точности в третьем параметре.

*Параметр format\_string имеет следующий вид:*

```
%[0 или более flags][field_width][.precision] conv_spec
```

flags (для m, mm, cm, e, df, di, sqm, sqcm, sqf, sqi, dd, gr, rad, cum, l, cumm, cumm, cuf, cui, cuy, gal):

```
(нет)      выравнивание вправо (по умолчанию);
-          выравнивание влево;
+          знак плюс задается явно;
(пробел)   вместо знака + ;
'*' 0      дополнительная точность отключена (по умолчанию);
'*' 1      дополнительная точность .5;
'*' 2      дополнительная точность .25;
'*' 3      дополнительная точность .1;
'*' 4      дополнительная точность .01;
'*' 5      дополнительная точность .5;
'*' 6      дополнительная точность .25.
```

flags (для m, mm, cm, df, di, sqm, sqcm, sqf, sqi, dd, fr, rad, cum, l, cumm, cumm, cuf, cui, cuy, gal):

```
'#'       не показывать 0 целых.
```

flags (для ffi, fdi, fi):

```
'0'       показывать 0 дюймов.
```

flags (для m, mm, cm, fdi, df, di, sqm, sqcm, sqf, sqi, dd, fr, rad, cum, l, cumm, cumm, cuf, cui, cuy, gal):

```
'~'       спрятать 0 целых (действует, если не специфицирован flag '#).
```

```
'^'       не изменять символ-разделитель дробной части и символ группирования цифр (если не указан, эти символы будут заменены согласно установкам текущей системы).
```

`field_width`: десятичное целое без знака; минимальное количество создаваемых символов.

`precision`: десятичное целое без знака; количество создаваемых дробных знаков.

conv\_sprec (спецификатор преобразования):

- e: экспоненциальный формат (метры);
- m: метры;
- mm: миллиметры;
- cm: сантиметры;
- ffi: футы и дюймы с долями;
- fdi: футы и дюймы с десятичными знаками;
- df: футы с десятичными знаками;
- fi: дюймы с долями;
- di: дюймы с десятичными знаками;

Для единиц измерения площадей:

- sqm: квадратные метры;
- sqcm: квадратные сантиметры;
- sqmm: квадратные миллиметры;
- sqf: квадратные футы;
- sqi: квадратные дюймы.

Для единиц измерения угловых величин:

- dd: градусы с десятичными знаками;
- dms: градусы, минуты, секунды;
- gr: градусы;
- rad: радианы;
- surv: геодезические единицы.

Для единиц измерения объемов:

- cum: кубические метры;
- l: литры;
- cusc: кубические сантиметры;
- cumm: кубические миллиметры;
- cuf: кубические футы;
- cui: кубические дюймы;
- cu: кубические ярды;
- gal: галлоны.

**Примеры:**

```

h = 23
nr = 0.345678
TEXT2 0, h, STR ("%m", nr) !0.346
TEXT2 0, h-1, STR ("%#10.2m", nr) !35
TEXT2 0, h-2, STR ("% .4cm", nr) !34.5678
TEXT2 0, h-3, STR ("%12.4cm", nr)!34.5678
TEXT2 0, h-4, STR ("% .6mm", nr)!345.678000
TEXT2 0, h-5, STR ("% +15e", nr)!+3.456780e-01
TEXT2 0, h-6, STR ("% ffi", nr) !1'-2"
TEXT2 0, h-7, STR ("%0.16ffi", nr) !1'-1 5/8"
TEXT2 0, h-8, STR ("% .3fdi", nr) ! 1'-1.609"
TEXT2 0, h-9, STR ("% -10.4df", nr) ! 1.1341'
TEXT2 0, h-10, STR ("%0.64fi", nr) !13 39/64"
TEXT2 0, h-11, STR ("% +12.4di", nr)!+13.6094"
TEXT2 0, h-12, STR ("%#.3sqm", nr) ! 346
TEXT2 0, h-13, STR ("% +sqcm", nr) !+3,456.78
TEXT2 0, h-14, STR ("% .2sqmm", nr)! 345,678.00
TEXT2 0, h-15, STR ("% -12sqf", nr) !3.72
TEXT2 0, h-16, STR ("%10sqi", nr)! 535.80

```

```

alpha = 88.657
TEXT2 0, h-17, STR ("% +10.3dd", alpha)!+88.657°
TEXT2 0, h-18, STR ("% .1dms", alpha)!88°39'
TEXT2 0, h-19, STR ("% .2dms", alpha) !88°39'25"
TEXT2 0, h-20, STR ("%10.4gr", alpha) ! 98.5078G
TEXT2 0, h-21, STR ("%rad", alpha) !1.55R
TEXT2 0, h-22, STR ("% .2surv", alpha) !N 1°20'35" E

```

**SPLIT**

**SPLIT** (string, format, variable1 [, variable2, ..., variablen])

Преобразует значение параметра string в один или несколько числовых или строковых элементов по заданному формату. Процесс останавливается при попытке считывания первого элемента, не соответствующего заданному формату. Возвращает количество успешно считанных значений. (*целое число*).

string: преобразуемая строка символов.

format: любая комбинация строковых констант и символов %s и %p -s. Элементы в строке должны соответствовать строковым константам в формате, %s обозначает любое строковое значение, ограниченное пробелами или знаками табуляции, %p обозначает любое числовое значение.

variable: имена переменных, в которых сохраняются результаты преобразования исходной строки.

*Пример:*

```
ss = "3 pieces 2x5 beam"
n = SPLIT (ss, "%n pieces %nx%n %s", num, ss1, size1, ss2, size2, name)
IF n = 6 THEN
    PRINT num, ss1, size1, ss2, size2, name      ! 3 балки 2 x 5
ELSE
    PRINT "ERROR"
ENDIF
```

## STW

**STW** (string\_expression)

Возвращает ширину (*действительное число*) строки (в метрах) с учетом текущего стиля. Ширина в метрах, при текущем масштабе равна STW (string\_expression) / 1000 \* A\_.

*Пример:*

abcd

```
DEFINE STYLE "own" "Monaco", 180000 / A_, 1, 0
SET STYLE "own"
string = "abcd"
width = STW (string) / 1000 * A_
n = REQUEST ("Height_of_style", "own", height)
height = height / 1000 * A_
text2 0,0, string
rect2 0,0, width, -height
```

## STRLEN

**STRLEN** (string\_expression)

возвращает длину (*целое число*) строки (количество символов).

## STRSTR

**STRSTR** (string\_expression1, string\_expression2)

Возвращает позицию (*целое число*) первого вхождения второй строки в первую. Если первая строка вообще не содержит вторую, то функция возвращает значение 0.

## STRSUB

**STRSUB** (string\_expression, start\_position, characters\_number)

Возвращает подстроку строкового параметра string\_expression, начинающуюся с позиции start\_position и длиной в characters\_number символов.

*Пример:*

```
ss = ""
  n = REQUEST ("Linear_dimension", "",ss)
  unit = ""
IF STRSTR (ss, "m") > 0 THEN unit = "m"
IF STRSTR (ss, "mm") > 0 THEN unit = "mm"
IF STRSTR (ss, "cm") > 0 THEN unit = "cm"
TEXT2 0, 0, STR (ss, a) + " " + unit !1.00 m
string = "Flowers.PICT"
len = STRLEN (string)
n = STRSTR (string, ".")
TEXT2 0, -1, STRSUB (string, 1, n - 1) !Flowers
TEXT2 0, -2, STRSUB (string, len - 4, 5) !.PICT
```





---

# УПРАВЛЯЮЩИЕ ПРЕДЛОЖЕНИЯ

*В этой главе описываются возможности языка GDL по управлению порядком вычисления команд, правилами обращения к подпрограммам, а также принципы работы с буфером для запоминания значений параметров с целью их дальнейшего использования. Здесь также объясняется, как использовать объекты в виде макровывозов и как выводить на экран результаты произведенных вычислений.*

## ПРЕДЛОЖЕНИЯ ПЕРЕДАЧИ УПРАВЛЕНИЯ

### FOR

**FOR** variable\_name = initial\_value **TO** end\_value [ **STEP** step\_value ]

Первое предложение цикла FOR. Если ключевое слово STEP вместе со step\_value опущены, то приращение переменной равно 1. В качестве переменной цикла нельзя использовать глобальную переменную.

*Пример:*

```
FOR I=1 TO 10 STEP 2
  PRINT I
NEXT I
```

### NEXT

**NEXT** variable\_name

Последнее предложение цикла FOR.

Переменная цикла variable\_name изменяется от initial\_value до end\_value с приращением step\_\_value при каждом выполнении тела цикла (предложений между FOR и NEXT). Если переменная цикла превышает значение end\_value, то происходит переход на предложение, следующее за NEXT.

**Примечание:** Изменение значения step\_value внутри цикла не оказывает никакого воздействия на выполнение цикла.

Следующие два фрагмента программы являются эквивалентными::

```
! Первый
A = B
1: IF C > 0 AND A > D OR C < 0 AND A < D THEN 2
  PRINT A
  A = A + C
  GOTO 1
2:
! Второй
FOR A = B TO D STEP C
  PRINT A
NEXT A
```

Как видно из предыдущего примера, значение `step_value = 0` приводит к бесконечному циклу.

После `FOR` допустимо только одно предложение `NEXT`. Вы можете выйти из цикла по предложению `GOTO` (или `IF ... GOTO`) и затем вернуться в него, однако нельзя войти в цикл в обход предложения `FOR`.

## DO

**DO**

```
[statement1
statement2
...
statementn]
```

**WHILE** condition

Предложения, заключенные между ключевыми словами `DO` и `WHILE` выполняются, до тех пор, пока значение выражения `condition` истинно.

Условие проверяется всякий раз после выполнения последовательности заданных в теле цикла предложений.

**WHILE** condition DO

```
[statement1
statement2
...
statementn]
```

**ENDWHILE**

Если значение выражения `condition` истинно, выполняется последовательность предложений, заключенных между ключевыми словами `DO` и `ENDWHILE`.

Условие проверяется перед каждым выполнением заданной последовательности предложений.

**REPEAT**

```
[statement1
```

```

statement2
...
statementn]

```

**UNTIL** condition

Предложения между ключевыми словами выполняются до тех пор, пока выражение condition не примет значение истинно. Условие проверяется всякий раз после выполнения последовательности заданных в теле цикла предложений.

*Пример:*

Следующие четыре фрагмента GDL-программы эквивалентны:

```

! Первый
FOR i = 1 TO 5 STEP 1
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
NEXT i
! Второй
i = 1
DO
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
    i = i + 1
WHILE i <= 5
! Третий
i = 1
WHILE i <= 5 DO
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
    i = i + 1
ENDWHILE
! Четвертый
i = 1
REPEAT
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
    i = i + 1
UNTIL i > 5

```

## IF

**IF** condition **THEN** label

**IF** condition **GOTO** label

**IF** condition **GOSUB** label

Предложение условного перехода. Если значение выражения condition равно 0, перехода нет, в противном случае происходит переход по метке label.

*Примеры.*

```
IF A THEN 28
```

```
IF I > J GOTO 200+I*J
```

```
IF I > 0 GOSUB 9000
```

```
IF condition THEN statement [ELSE statement]
```

или

```
IF condition THEN [statement1  
statement2
```

```
...
```

```
statementn]
```

```
[ELSE
```

```
statementn+1
```

```
statementn2
```

```
...
```

```
statementn+m]
```

```
ENDIF
```

Если после ключевых слов THEN и/или ELSE записана только одна команда в той же строке, что и ключевое слово, то нет необходимости в ключевом слове ENDIF. Команда после THEN или ELSE, записанная в той же строке, подразумевает наличие после нее ENDIF.

Если после ключевого слова THEN есть новая строка, то следующие команды (все команды до ключевого слова ELSE или ENDIF) будут выполняться, когда выражение condition истинно (отлично от нуля). В противном случае, будут выполняться команды, следующие после ELSE. Если ключевое слово ELSE отсутствует, то будут выполняться команды, следующие после ENDIF.

*Пример:*

```

IF a = b THEN height = 5 ELSE height = 7
IF needdoors THEN
    CALL "door_macro" PARAMETERS
    ADDX a
ENDIF
IF simple THEN
    HOTSPOT2 0, 0
    RECT2 a, 0, 0, b
ELSE PROJECT2 3, 270, 1
IF name = "Sphere" THEN
    ADDY b
    SPHERE 1
ELSE
    ROTX 90
    TEXT 0.002, 0, name
ENDIF

```

## **GOTO**

**GOTO** label

Предложение безусловного перехода. Происходит переход на предложение с меткой label (числовой или строковой). Выражение метки label, содержащее переменные, может замедлить интерпретацию в связи с необходимостью вычисления адреса перехода.

*Пример:*

```
GOTO K+2
```

## **GOSUB**

**GOSUB** label

Обращение к внутренней подпрограмме. Метка label является точкой входа в подпрограмму. Значение label может быть числового или строкового типа. Выражение метки label, содержащее переменные, может замедлить интерпретацию в связи с необходимостью вычисления адреса перехода.

## **RETURN**

**RETURN**

Происходит выход из внутренней подпрограммы.

## END / EXIT

**END / EXIT** [v1, v2, ..., vn]

Конец GDL-скрипта. Программа завершает работу и происходит возврат на предыдущий уровень. В файле GDL можно использовать несколько предложений END или EXIT. Если указан факультативный список значений, то текущий скрипт возвращает эти значения в вызывающую программу.

См. описание приема возвращаемых параметров в "CALL" на стр. 218.

## BREAKPOINT

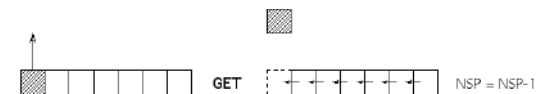
**BREAKPOINT** expression

Эта команда позволяет определить точки останова в GDL-скрипте. GDL-скрипте. Отладчик GDL будет по этой команде останавливать выполнение скрипта, если значение параметра (числового выражения) истинно (равно 1) и установлен параметр активирования точек останова в отладчике. При нормальном режиме выполнения программы интерпретатор GDL просто пропускает эти команды.

## МАНИПУЛИРОВАНИЕ БУФЕРОМ ПАРАМЕТРОВ

**Буфер параметров** - это встроенная структура данных, которая может использоваться, если, например, есть необходимость временно изменить некоторое "стандартное" значение (координата, например) с последующим ее восстановлением. Он также используется, если необходимо сохранить текущие значения переменных.

Буфер параметров - это бесконечно длинный массив, в котором можно сохранять числовые значения, используя команду PUT. Команда PUT помещает заданные значения в конец буфера. Эти значения могут быть извлечены позже (команды GET, USE) в той же последовательности, в которой они были помещены в буфер (так первое сохраненное значение будет первым выбираемым из буфера значением). Команды GET (n) и USE (n) эквивалентны и выдают в качестве своего результата n значений, разделенных запятыми. Таким образом, они могут использоваться в любом списке параметров GDL, где необходимо задать n значений.



## **PUT**

**PUT** expression [ , expression, ...]

Сохраняет заданные значения в указанном порядке во внутреннем буфере параметров.

## **GET**

**GET** (n)

Извлекает следующие n значений из внутреннего буфера параметров с их удалением из него.

## **USE**

**USE** (n)

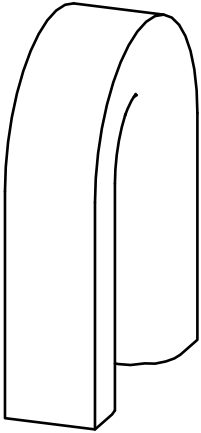
Извлекает следующие n значений из внутреннего буфера параметров без их удаления. Функция GET, следующая за USE, извлекает те же параметры.

## **NSP**

**NSP**

Возвращает количество параметров в буфере.

Пример использования буфера параметров:



```
R=2: B=6: C=4: D=10  
N=12
```

```
S=180/N  
FOR T=0 TO 180 STEP S  
  PUT R+R*COS(T), C-R*SIN(T), 1  
NEXT T
```

```
FOR I=1 TO 2  
  EXTRUDE 3+NSP/3, 0,0,D, 1+16,  
    0, B, 0,  
    2*R, B, 0,  
    USE(NSP),  
    0, B, 0  
  MULY -1  
NEXT I  
DEL 1  
ADDZ D  
REVOLVE 3+NSP/3, 180, 0,  
  0, B, 0,  
  2*R, B, 0,  
  GET(NSP),  
  0, B, 0
```



*Полное описание без использования буфера:*

R=2: B=6: C=4: D=10

```

FOR I=1 TO 2
  EXTRUDE 16, 0,0,D, 1+16,
    0, B, 0,
    2*R, B, 0,
    2*R, C, 1,
    R+R*COS(15), C-R*SIN(15), 1,
    R+R*COS(30), C-R*SIN(30), 1,
    R+R*COS(45), C-R*SIN(45), 1,
    R+R*COS(60), C-R*SIN(50), 1,
    R+R*COS(75), C-R*SIN(75), 1,
    R+R*COS(90), C-R*SIN(90), 1,
    R+R*COS(105), C-R*SIN(105), 1,
    R+R*COS(120), C-R*SIN(120), 1,
    R+R*COS(135), C-R*SIN(135), 1,
    R+R*COS(150), C-R*SIN(150), 1,
    R+R*COS(165), C-R*SIN(165), 1,
    0, B, 1,
    0, B, 0
  MULLY -1
NEXT I
DEL 1
ADDZ D
REVOLVE 16, 180, 0,
  0, B, 0,
  2*R, B, 0,
  2*R, C, 1,
  R+R*COS(15), C-R*SIN(15), 1,
  R+R*COS(30), C-R*SIN(30), 1,
  R+R*COS(45), C-R*SIN(45), 1,
  R+R*COS(60), C-R*SIN(50), 1,
  R+R*COS(75), C-R*SIN(75), 1,
  R+R*COS(90), C-R*SIN(90), 1,
  R+R*COS(105), C-R*SIN(105), 1,
  R+R*COS(120), C-R*SIN(120), 1,
  R+R*COS(135), C-R*SIN(135), 1,
  R+R*COS(150), C-R*SIN(150), 1,
  R+R*COS(165), C-R*SIN(165), 1,
  0, B, 1,
  0, B, 0

```

## МАКРООБЪЕКТЫ

Хотя любые 3D-объекты можно разбить на сложные или примитивные элементы, однако иногда желательно определить некоторые из них в виде самостоятельных элементов. Такие самостоятельно определенные элементы называются макросами.

### CALL

```
CALL macro_name_string [,parameter_list]
```

```
CALL macro_name_string [,] PARAMETERS [name1=value1 , ... namen=valuen] [[,]
RETURNED_PARAMETERS r1, r2, ...]
```

```
CALL macro_name_string [,] PARAMETERS ALL [name1=value1 , ...
namen=valuen] [[,] [RETURNED_PARAMETERS r1, r2, ...]
```

```
CALL macro_name_string [,] PARAMETERS value1 or DEFAULT [, ... valuen or DEFAULT]
```

Имена макросов не могут превышать 31 символ.

Макроимя может быть константой или переменной строкового типа, или параметром. Нельзя использовать операции над строками применительно к макровызову также как и применительно к макроимени.

**Внимание:** Если в качестве макроимен используются параметры или переменные строкового типа, то вызываемый макрос не будет включаться в архивный проект, даже, если установлен маркер *Включить все элементы загруженных библиотек*.

Макроимя должно заключаться в кавычки (“ ’ ` , ; , ; ” ’ ; “ ; ’), если оно не согласуется с правилами определения идентификаторов. Макроимя соответствует определению идентификатора, если оно начинается с буквы, символа ‘\_’ или ‘-’ и содержит только буквы, цифры и символы ‘\_’ и ‘~’. В противном случае в предложении CALL следует использовать кавычки, причем обе кавычки должны быть одинаковыми и они не должны использоваться в имени макроса.

В качестве команды вызова макроса можно использовать само макроимя (без ключевого слова CALL) в следующем формате:

```
macro_name [parameter_list]
```

```
macro_name PARAMETERS [name1=value1, ... namen=valuen]
```

```
macro_name PARAMETERS ALL
```

Макровызов первого типа может использоваться как с простыми текстовыми файлами GDL, так и с произвольным библиотечным элементом, при условии, что его список параметров состоит только из однобуквенных параметров (A .. Z). Такая форма макровызова может использоваться для совместимости с предыдущими версиями, но мы рекомендуем использовать макровызов второго типа. Смысл списка параметров следующий: значение параметра A будет первым значением в списке, значение параметра B - вторым, и т.д. Если макрос (библиотечного элемента) не имеет однобуквенного параметра, соответствующего значению в списке, то это значение будет пропущено, процесс интерпретации будет продолжен, а программа выдаст предупреждение. В списке параметров не допускаются никакие выражения строкового типа.

Макровызовы второго типа могут использоваться только с полностью описанными библиотечными элементами или обычными текстовыми файлами GDL. После ключевого слова PARAMETERS необходимо перечислить в произвольном порядке имена параметров вызываемого макроса, и для каждого из них указать значение после знака ‘=’. Разрешается использовать выражения

строкового типа, но надо внимательно следить, чтобы строковое значение присваивалось только параметрам строкового типа вызываемого макроса. Для параметров типа массив должен быть задан полный массив значений такой же размерности. Если в списке параметров присутствует имя, которого нет среди параметров вызываемого макроса, то после вызова макроса будет выдано сообщение об ошибке. Если в макровывозе отсутствует необходимый параметр, то ему будет присвоено исходное значение по умолчанию, как оно определено в библиотечном элементе, вызванном как макрос.

Макровывозы третьего типа могут использоваться только с полностью описанными библиотечными элементами. В этом случае нет необходимости специфицировать параметры поодиночке; все параметры вызывающего скрипта передаются в вызываемый макрос. Для параметра макроса, которого нет в вызывающем скрипте, используется значение по умолчанию. Если значения параметров задаются раздельно, то они заменяют значения, поступающие из вызывающего скрипта, или параметры вызываемого макроса выбираются по умолчанию. В этом случае макрос также может возвращать параметры. В вызывающем скрипте возвращаемые значения могут быть собраны с помощью ключевого слова `RETURNED_PARAMETERS`, за которым следует список переменных. Возвращаемые значения запоминаются в этих переменных в том порядке, в котором возвращаются из макроса. Количество и тип переменных, определяемых в вызывающем скрипте и вызываемом макросе не обязательно должны совпадать. Если в вызывающем скрипте больше переменных, то "лишним" переменным присваивается значение 0 (целое). Совместимость типов не проверяется: переменным вызывающего скрипта устанавливается тип соответствующих переменных вызываемого макроса. Если одна из переменных вызывающего скрипта является динамическим массивом, то все последующие значения запоминаются в этом массиве.

*См описание синтаксиса возвращаемых параметров в "END / EXIT" на стр. 214.*

Макрос GDL имеет свою рабочую среду, которая зависит от характера его вызова. Значения, установленные по командам `MODEL`, `RADIUS`, `RESOL`, `TOLER`, `PEN`, `LINE_TYPE`, `MATERIAL`, `FILL`, `STYLE`, `SHADOW` а также текущее преобразование координат являются доступными в макросе. Они могут в нем использоваться и изменяться, однако изменения носят локальный характер. То есть эти значения восстанавливаются при выходе из макроса.

Наличие параметров в макровывозе свидетельствует о неявном присвоении значений переменным на уровне макроса.

Параметры A и B обычно используются для изменения размеров объектов.

Макровывозы четвертого типа также могут использоваться только с полностью описанными библиотечными элементами. В этом случае фактические значения параметров должны быть специфицированы раздельно и в том порядке, в котором они определены в вызываемом библиотечном элементе: значения не должны отсутствовать за исключением конца списка. Использование ключевого слова `DEFAULT` вместо фактического значения параметра означает, что фактическим значением будет значение по умолчанию, запомненное в библиотечном элементе. Для отсутствующих значений значения по умолчанию применяются автоматически (количество фактических значений  $m$  может быть меньше количества параметров). При интерпретации макроса этого типа нет необходимости находить параметры по именам для присвоения им фактических значений. Хотя этот вариант менее удобный для использования, чем предыдущие, однако при этом достигается большая производительность.

*Примеры:*

```
CALL "leg" 2, , 5 ! A = 2, B = 0, C = 5
leg 2, , 5
CALL "door-1" PARAMETERS height = 2, a = 25.5,
    name = "Director"
CALL "door-1" PARAMETERS
    ! для параметров используются значения по умолчанию
"door-1" PARAMETERS
```

**Резюме:** Если у Вас нет необходимости в параметрах с длинными именами или параметрах строкового типа, то может быть вполне достаточным использование GDL текстового типа. Такой тип GDL может быть вызван только при помощи макровывода первого типа, так как он не имеет списка параметров. С другой стороны, если Вы не желаете ограничивать имена макропараметров буквами от A до Z, или хотите включить в список параметров символьные строки, Ваш макрос должен быть библиотечным элементом и вызываться в соответствии с GDL-синтаксисом макровывода второго типа.

## ПРЕДЛОЖЕНИЕ ВЫВОДА НА ЭКРАН

### PRINT

```
PRINT expression [, expression, ...]
```

Выводит на экран все свои аргументы в специальном диалоговом окне. Аргументами могут быть выражения строкового или числового типа в любом количестве и в любой последовательности, разделенные запятыми.

*Примеры:*

```
PRINT ""Значение переменной цикла:", I
PRINT J, K-3*L
PRINT "Начало интерпретации"
PRINT a * SIN (alpha) + b * COS (alpha)
PRINT "Значения параметров: ", "a = ", a, ", b = ", b
PRINT name + STR ("%m", i) + "." + ext
```

## ОПЕРАЦИИ ДЛЯ РАБОТЫ С ФАЙЛАМИ

Приведенные ниже команды позволяют открывать внешние файлы для чтения/записи и манипулировать ими посредством ввода/вывода значений в/из GDL-скрипта. В этот процесс обязательно вовлекаются специальные расширения ArchiCAD. Текстовые файлы могут обрабатываться посредством расширения “TEXT GDL I/O”. Другие независимые разработчики могут создавать расширения иных типов.

См. также *“Расширение GDL Text I/O”* в *“Разное”*.

## OPEN

**OPEN** (*filter*, *filename*, *parameter\_string*)

*filter*: строка, указывающая имя существующего расширения.

*filename*: строка, указывающая имя файла.

*parameter\_string*: строка; содержит специфические символы-разделители, режимы открытия файла и другие параметры. Эти значения интерпретируются вызываемым расширением.

Приводит к открытию файла заданным образом. Функция возвращает целое положительное значение, идентифицирующее этот файл. Это значение, номер канала, в дальнейшем может использоваться в качестве ссылочного номера файла. Для включения этого ссылаемого файла в архив проекта используйте команду “FILE\_DEPENDENCE "name1" [, "name2", ...]” с указанием имени файла.

## INPUT

**INPUT** (*channel*, *recordID*, *fieldID*, *variable1* [, *variable2*, ...])

*recordID*, *fieldID*: параметры строкового или числового типа, задающие исходную позицию, начиная с которой начинается чтение из файла; содержание этих параметров интерпретируется расширением.

Количество заданных параметров *variable* определяет сколько значений должно быть считано из файла, определяемого значением *channel* (ссылочный номер файла), начиная с указанной исходной позиции. В списке параметров должен быть, по крайней мере, один параметр *variable*. Эта функция присваивает выбираемые из файла значения параметрам *variable* согласно их расположению в списке. Значения могут быть числового или строкового типа, независимо от того какой тип имеют параметры.

Функция возвращает число успешно прочитанных значений. Если встречается признак конца файла, то возвращается -1.

## VARTYPE

**VARTYPE** (*expression*)

Возвращает 1, если *expression* имеет числовой тип, и 2, если *expression* имеет строковый тип.

Эта функция оказывается полезной при чтении значений в переменные с помощью команды **INPUT**, которая может изменить тип переменных согласно текущим значениям.

Тип этих переменных не проверяется при компиляции.

## OUTPUT

**OUTPUT** *channel*, *recordID*, *fieldID*, *expression1* [, *expression2*, ...]

*recordID*, *fieldID*: параметры строкового или числового типа, задающие исходную позицию, начиная с которой начинается запись в файл. Содержание этих параметров интерпретируется расширением.

Количество заданных выражений `expression` определяет, сколько значений должно быть записано в файл, определяемый значением `channel` (ссылочный номер файла), начиная с указанной исходной позиции. В списке параметров должно быть, по крайней мере, одно выражение `expression`. Тип записываемых значений совпадает с типом значений выражений.

## **CLOSE**

**CLOSE** `channel`

Приводит к закрытию файла, определяемого значением `channel`.

---

# РАЗНОЕ

*GDL также может оперировать операциями над внешними файлами с помощью специальных приложений, расширяющих функциональные возможности ArchiCAD. Эти команды описываются в этой главе и иллюстрируются с помощью примеров.*

## ГЛОБАЛЬНЫЕ ПЕРЕМЕННЫЕ

Глобальные переменные дают возможность запоминать специальные значения модели. При этом доступ к таким значениям можно производить из макросов GDL. Так, например, при определении окна Вы имеете возможность получить параметры стены, в которую оно встроено. Глобальные параметры не помещаются в стек при вызове макросов.

Для окон, дверей, выносных надписей и спецификаций библиотечных элементов имеется еще одна возможность взаимодействовать с ArchiCAD с помощью факультативных параметров, которые имеют фиксированные имена. Эти параметры, если они присутствуют в списке параметров библиотечного элемента, устанавливаются с помощью ArchiCAD. См. список параметров с фиксированными именами и другую дополнительную информация в документации по основной библиотеке ArchiCAD в <http://www.graphisoft.com/support/developer/documentation/LibraryDevDoc/10>.

## Общие параметры рабочей среды

---

<b>GLOB_SCRIPT_TYPE</b>	<b>T~</b>	<b>тип текущего скрипта</b>
<i>1 - скрипт спецификаций, 2- 2D-скрипт, 3-3D-скрипт, 4 - скрипт интерфейса пользователя, 5 - скрипт параметров, 6 - основной скрипт</i>		
<b>GLOB_CONTEXT</b>		<b>контекст использования</b>
<i>1 - редактор библиотечного элемента, 2 - план этажа, 3 - 3D-вид, 4 - разрез/фасад, 5 - диалог установки параметров, 6 - список сметы, 7 - чертеж детали, 8 - макет, 22 - режим обратной связи с плана этажа, 23 - режим обратной связи из 3D-вида, 24 - режим обратной связи из разреза/фасада, 28 - режим обратной связи из макета, 43 - генерация, как оператор из 3D-вида, 44 - генерация, как оператор из разреза/фасада, 46 - генерация, как оператор из сметы.</i>		
<b>GLOB_SCALE</b>	<b>A_</b>	<b>масштаб чертежа</b>
<i>согласно текущему окну</i>		
<b>GLOB_DRAWING_BGD_PEN</b>		<b>перо цвета фона чертежа</b>
<i>перо текущей панели, наиболее подходящие для цвета фона текущего окна</i>		
<b>GLOB_NORTH_DIR</b>	<b>U~</b>	<b>направление на север проекта</b>
<i>определяется относительно установленной по умолчанию системы координат проекта в соответствии с установками параметров солнца</i>		
<b>GLOB_WORLD_ORIGO_OFFSET</b>		
<i>Расположение начала координат проекта относительно начала мировой систему координат. См. "Пример, иллюстрирующий использование глобальных переменных GLOB_WORLD_ORIGO_...:" на стр. 242.</i>		
<b>GLOB_MODPAR_NAME</b>		<b>имя последнего измененного параметра</b>
<i>в диалоговом окне установки параметров</i>		

---

---

**GLOB\_UI\_BUTTON\_ID** ID кнопки, нажатой на странице интерфейса пользователя

*или 0, если последнее выполненное действие было не нажатие кнопки с  $id$ .*

**GLOB\_CUTPLANES\_INFO** [ 4 ] -

*массив с 4 значениями: 1: высота плоскости сечения, 2: верхний уровень плоскости сечения, 3: нижний уровень плоскости сечения, 4: абсолютный предел воспроизведения, в локальной системе координат библиотечного элемента..*

*См. подробности в описании диалогового окна ArchiCAD Параметры плоскости сечения плана этажа.*

---

## Информация об этажах

**GLOB\_HSTORY\_ELEV** **B** **возвышение исходного этажа**

*исходным является тот этаж, на котором размещаются объекты.*

**GLOB\_HSTORY\_HEIGHT** **Q** **высота исходного этажа**

*исходным является тот этаж, на котором размещаются объекты.*

**GLOB\_CSTORY\_ELEV** **Q~** **возвышение текущего этажа**

*текущим является тот этаж, изображение которого выводится в окне плана этажа в текущий момент.*

**GLOB\_CSTORY\_HEIGHT** **R~** **высота текущего этажа**

*текущим является тот этаж, изображение которого выводится в окне плана этажа в текущий момент.*

**GLOB\_CH\_STORY\_DIST** **S~** **расположение текущего этажа относительно исходного**

*текущим является тот этаж, изображение которого выводится в окне плана этажа в текущий момент.*

---

## Информация о съемке

**GLOB\_FRAME\_NR** **N** **номер текущего кадра в анимации**

*действителен только для анимаций, для всех остальных типов изображений равен 0.*

**GLOB\_FIRST\_FRAME** **O** **индекс первого кадра траектории съемки**

*действителен только для анимаций, для всех остальных типов изображений равен 0.*

**GLOB\_LAST\_FRAME** **P** **индекс последнего кадра траектории съемки**

*действителен только для анимаций, для всех остальных типов изображений равен 0*

**GLOB\_EYEPOS\_X** **K~** **текущее положение камеры (по оси x)**

*действителен только для анимаций, для всех остальных типов изображений равен 0*

**GLOB\_EYEPOS\_Y** **L~** **текущее положение камеры (по оси y)**

*действителен только в перспективной проекции как для анимаций, так и для других изображений*

**GLOB\_EYEPOS\_Z** **M~** **текущее положение камеры (по оси z)**

*действителен толь в перспективной проекции как для анимаций, так и для других изображений*

**GLOB\_TARGPOS\_X** **N~** **текущее расположение цели (по оси x)**

*действителен толь в перспективной проекции как для анимаций, так и для других изображений*



<b>GLOB_TARGPOS_Y</b>	<b>O~</b>	<b>текущее расположение цели (по оси y)</b> <i>действителен толь в перспективной проекции как для анимаций, так и для других изображений</i>
<b>GLOB_TARGPOS_Z</b>	<b>P~</b>	<b>текущее расположение цели (по оси z)</b> <i>действителен толь в перспективной проекции как для анимаций, так и для других изображений</i>
<b>GLOB_SUN_AZIMUTH</b>		<b>азимут солнца</b> <i>согласно параметрам диалогового окна Солнце</i>
<b>GLOB_SUN_ALTITUDE</b>		<b>высота солнца</b> <i>согласно параметрам диалогового окна Солнце</i>

## Общие параметры элементов

<b>GLOB_LAYER</b>		<b>слой элемента</b> <i>название слоя, на котором расположен элемент</i>
<b>GLOB_ID</b>		<b>пользовательский ID элемента</b> <i>ID, заданный в диалоге установки параметров элемента</i>
<b>GLOB_INTID</b>		<b>внутренний ID элемента</b> <i>внутренний уникальный идентификатор ID, созданный программой (не контролируется пользователем)</i>
<b>GLOB_INTGUID</b>		<b>внутренний GUID элемента, внутренний GUID, созданный программой</b> <i>(не контролируется пользователем)</i>
<b>GLOB_ELEVATION</b>	<b>J_</b>	<b>возвышение основания элемента</b> <i>относительно исходного этажа (за исключением дверей и окон: высота подоконника/порога, согласно текущих установок параметров)</i>
<b>GLOB_ELEM_TYPE</b>		<b>тип элемента, для выносных надписей и объектов спецификаций содержит тип родительского элемента</b> <i>0 - нет (индивидуальная выносная надпись), 1 - объект, 2 - источник света, 3 - окно, 4 - дверь, 5 - стена, 6 - колонна, 7 - перекрытие, 8 - крыша, 9 - заштрихованная область, 10-3D-сетка, 11 - зона, 12 - балка</i>

## Параметры объектов, источников света, дверей и окон

<b>SYMB_LINETYPE</b>		<b>тип линии библиотечного элемента</b> <i>применяется в качестве установленного по умолчанию типа линии 2D-символа</i>
<b>SYMB_FILL</b>		<b>образец штриховки библиотечного элемента</b> <i>применяется на секущих поверхностях библиотечных элементов в окнах разрезов/фасадов</i>
<b>SYMB_FILL_PEN</b>		<b>перо штриховки библиотечного элемента</b> <i>применяется на секущих поверхностях библиотечных элементов в окнах разрезов/фасадов</i>
<b>SYMB_FBGD_PEN</b>		<b>перо фона штриховки библиотечного элемента</b> <i>применяется на секущих поверхностях библиотечных элементов в окнах разрезов/фасадов</i>
<b>SYMB_SECT_PEN</b>		<b>перо библиотечного элемента в разрезе</b> <i>применяется в качестве контура секущих поверхностей библиотечных элементов в окнах разрезов/фасадов</i>
<b>SYMB_VIEW_PEN</b>	<b>L_</b>	<b>перо по умолчанию библиотечного элемента</b> <i>применяется для всех ребер в 3D-окне и для ребер в окнах разрезов/фасадов</i>
<b>SYMB_MAT</b>	<b>M_</b>	<b>покрытие по умолчанию библиотечного элемента</b>
<b>SYMB_POS_X</b>	<b>X~</b>	<b>расположение библиотечного элемента (координата x)</b> <i>относительно начала проектных координат (за исключением дверей, окон и концов стен: расположение определяется относительно начальной точки стены, которая их содержит)</i>
<b>SYMB_POS_Y</b>	<b>Y~</b>	<b>расположение библиотечного элемента (координата y)</b> <i>относительно начала проектных координат (за исключением окон, дверей и концов стен: относительно начальной точки содержащей их стены) Примечание: см. “Двери и окна” на стр. 257 для ознакомления с ориентацией осей Y и Z.</i>
<b>SYMB_POS_Z</b>	<b>Z~</b>	<b>расположение библиотечного элемента (z)</b> <i>относительно начала проектных координат (за исключением дверей, окон и концов стен: расположение определяется относительно начальной точки стены, которая их содержит) Примечание: см. “Двери и окна” на стр. 257 для ознакомления с ориентацией осей Y и Z</i>

## Параметры объектов и источников света

<b>SYMB_ROTANGLE</b>	<b>W~</b>	<b>угол поворота библиотечного элемента</b> <i>поворот, заданный в диалоге установки параметров числовым значением величины угла; производится вокруг текущей точки привязки элемента</i>
<b>SYMB_MIRRORED</b>	<b>V~</b>	<b>зеркальное отражение библиотечного элемента</b> <i>0 - нет, 1 - отражается (отражение осуществляется относительно текущей точки привязки) Всегда 0 для концов стен, за исключением случая, когда начало локальных координат находится в непрямоугольной вершине трапецидального многоугольника стены.</i>

## Параметры объектов, источников света, дверей и окон - доступны только в смете и выносных надписях

**SYMB\_A\_SIZE** номинальная длина/ширина библиотечного элемента

*длина объекта/источника света, ширина двери/окна фиксированные параметры).*

**SYMB\_B\_SIZE** номинальная ширина/высота библиотечного элемента

*ширина объекта/источника света, высота окна/двери (фиксированные параметры)*

## Параметры объектов и источников света - доступны только в смете и выносных надписях

**SYMB\_Z\_SIZE** номинальная высота библиотечного элемента

*если параметр пользователя поименован в формате zzyzx, то значение этого параметра будет использоваться в качестве номинальной высоты, иначе 0*

## Параметры дверей, окон и концов стен

**WIDO\_REVEAL\_ON** наличие четверти оконного/дверного проема

*0 - четверть отсутствует, 1 - четверть присутствует*

**WIDO\_SILL** **K\_** подоконник/порог оконного/дверного проема

*глубина четверти, установленная в панели Четверть диалогового окна установки параметров окна/двери для криволинейных стен: в радиальном направлении относительно угла проема номинального размера*

**WIDO\_SILL\_HEIGHT** номинальная высота подоконника/порога оконного/дверного проема

**WIDO\_RSIDE\_SILL\_HEIGHT** высота подоконника/порога оконного/дверного проема со стороны четверти

**WIDO\_OPRSIDE\_SILL\_HEIGHT** высота подоконника/порога оконного/дверного проема со стороны,

**WIDO\_RIGHT\_JAMB** **B~** ширина четверти оконного/дверного проема с правой стороны

*согласно параметрам панели Четверть диалогового окна установки параметров окна/двери*

**WIDO\_LEFT\_JAMB** ширина четверти оконного/дверного проема с левой стороны

*согласно параметрам панели Четверть диалогового окна установки параметров окна/двери*

**WIDO\_THRES\_DEPTH** **C~** высота нижней четверти оконного/дверного проема

*согласно параметрам панели Четверть диалогового окна установки параметров окна/двери*

**WIDO\_HEAD\_DEPTH** **D~** высота верхней четверти оконного/дверного проема

*согласно параметрам панели Четверть диалогового окна установки параметров окна/двери*

**WIDO\_HEAD\_HEIGHT** номинальная высота верхней части оконного/дверного проема

**WIDO\_RSIDE\_HEAD\_HEIGHT** высота верхней части оконного/дверного проема со стороны четверти

**WIDO\_OPRSIDE\_HEAD\_HEIGHT** высота верхней части оконного/дверного проема со стороны,

<b>WIDO_REVEAL_SIDE</b>	<b>E~</b>	<b>Сторона четверти противоположна стороне открывания</b> <i>1-да, 0-нет - при размещении элемента по умолчанию принимается 0 для окон, 1 для дверей</i>
<b>WIDO_FRAME_THICKNESS</b>	<b>F~</b>	<b>толщина оконной/дверной коробки</b> <i>при зеркальном отражении окно/дверь автоматически повторно размещается с использованием этого параметра.</i>
<b>WIDO_POSITION</b>	<b>H~</b>	<b>смещение окна/двери</b> <i>угол или расстояние между осью проема или конца стены и вектором нормали, проведенным к началу стены</i>
<b>WIDO_ORIENTATION</b>		<b>ориентация открывания двери/окна</b> <i>влево/вправо- хорошо действует, когда дверной/оконный проем создан в соответствии с локальными стандартами</i>
<b>WIDO_MARKER_TXT</b>		<b>текст маркера окна/двери</b> <i>устанавливается в диалоге определения размеров окон/дверей, вызываемом из диалога установки параметров окон/дверей</i>
<b>WIDO_SUBFL_THICKNESS</b>		<b>толщина пола (корректировка подоконника/порога)</b> <i>устанавливается в панели Параметры диалога установки параметров окон/дверей</i>
<b>WIDO_PREFIX</b>		<b>префикс высоты подоконника/порога окна/двери</b> <i>устанавливается в диалоге определения размеров окон/дверей, вызываемом из диалога установки параметров окон/дверей</i>
<b>WIDO_CUSTOM_MARKER</b>		<b>переключатель для специального маркера окна/двери</b> <i>1-параметры могут использоваться в 2D-скрипте в то время как автоматическое проставление размеров отсутствует</i>
<b>WIDO_ORIG_DIST</b>	<b>R_</b>	<b>расстояние от начала локальных координат до центра криволинейной стены</b> <i>для дугообразных стен: расстояние от начала локальных координат до центра криволинейной стены для прямолинейных стен - 0.</i>
<b>WIDO_PWALL_INSET</b>		<b>Вставка парапетной стены</b>

## Параметры окон и дверей - доступны только в смете и выносных надписях

<b>WIDO_RSIDE_WIDTH</b>	<b>Ширина дверного/оконного проема со стороны четверти</b>
<b>WIDO_OPRSIDE_WIDTH</b>	<b>Ширина дверного/оконного проема со стороны, противоположной четверти</b>
<b>WIDO_RSIDE_HEIGHT</b>	<b>Высота дверного/оконного проема со стороны четверти</b>
<b>WIDO_OPRSIDE_HEIGHT</b>	<b>Высота дверного/оконного проема со стороны, противоположной четверти</b>
<b>WIDO_RSIDE_SURF</b>	<b>Поверхность дверного/оконного проема со стороны четверти</b>
<b>WIDO_OPRSIDE_SURF</b>	<b>Площадь поверхности дверного/оконного проема со стороны, противоположной четверти</b>
<b>WIDO_N_RSIDE_WIDTH</b>	<b>Номинальная ширина дверного/оконного проема со стороны четверти</b>
<b>WIDO_N_OPRSIDE_WIDTH</b>	<b>Номинальная ширина дверного/оконного проема со стороны, противоположной четверти</b>
<b>WIDO_N_RSIDE_HEIGHT</b>	<b>Номинальная высота дверного/оконного проема со стороны четверти</b>
<b>WIDO_N_OPRSIDE_HEIGHT</b>	<b>Номинальная высота дверного/оконного проема со стороны, противоположной четверти</b>

<b>WIDO_N_RSIDE_SURF</b>	<b>Номинальная площадь поверхности дверного/оконного проема со стороны четверти</b>
<b>WIDO_N_OPRSIDE_SURF</b>	<b>Номинальная площадь поверхности дверного/оконного проема со стороны, противоположной четверти</b>
<b>WIDO_VOLUME</b>	<b>Объем дверного/оконного проема</b>
<b>WIDO_GROSS_SURFACE</b>	<b>Номинальная площадь поверхности дверного/оконного проема</b>
<b>WIDO_GROSS_VOLUME</b>	<b>Номинальный объем дверного/оконного проема</b>

## **Параметры источников света - доступны только в смете и выносных надписях**

<b>LIGHT_ON</b>	<b>включение источника света</b> <i>0 - источник выключен, 1 - источник включен: устанавливается в диалоге установки параметров источника света (параметр фиксированный)</i>
<b>LIGHT_RED</b>	<b>красная составляющая цвета источника света</b> <i>устанавливается в диалоге установки параметров источников света (параметр фиксированный)</i>
<b>LIGHT_GREEN</b>	<b>зеленая составляющая цвета источника света</b> <i>устанавливается в диалоге установки параметров источников света (параметр фиксированный)</i>
<b>LIGHT_BLUE</b>	<b>синяя составляющая цвета источника света</b> <i>устанавливается в диалоге установки параметров источников света (параметр фиксированный)</i>
<b>LIGHT_INTENSITY</b>	<b>интенсивность источника света</b> <i>устанавливается в диалоге установки параметров источников света (параметр фиксированный)</i>

## Параметры выносных надписей

<b>LABEL_POSITION</b>	<b>расположение выносной надписи</b> <i>массив [3][2], содержащий координаты трех точек, определяющих расположение выносной надписи</i>
<b>LABEL_CUSTOM_ARROW</b>	<b>использование маркера символа</b> <i>1 - отмечена кнопка Использовать маркер символа, 0 - в противном случае</i>
<b>LABEL_ARROW_PEN</b>	<b>перо линии-выноски согласно диалогу установки параметров</b>
<b>LABEL_ARROWHEAD_PEN</b>	<b>перо стрелки согласно диалогу установки параметров</b>
<b>LABEL_FONT_NAME</b>	<b>имя шрифта согласно диалогу установки параметров</b>
<b>LABEL_TEXT_SIZE</b>	<b>размер текста согласно диалогу установки параметров</b>
<b>LABEL_TEXT_PEN</b>	<b>перо текста согласно диалогу установки параметров</b>
<b>LABEL_FONT_STYLE</b>	<b>стиль шрифта согласно диалогу установки параметров</b> <i>0-обычный, 1-жирный, 2-курсив</i>
<b>LABEL_FRAME_ON</b>	<b>наличие рамки</b> <i>1 - если отмечен маркер Текст в рамке, 0 в противном случае</i>
<b>LABEL_ANCHOR_POS</b>	<b>место соединения линии-выноски с надписью</b> <i>0 - посередине, 1 - вверху, 2 - внизу согласно диалогу установки параметров</i>
<b>LABEL_ROTANGLE</b>	<b>угол поворота согласно диалогу установки параметров</b>
<b>LABEL_TEXT_ALIGN</b>	<b>выравнивание текста согласно диалогу установки параметров</b> <i>1: выравнивание влево, 2: выравнивание по центру, 3: выравнивание вправо, 4: выравнивание по ширине</i>
<b>LABEL_TEXT_LEADING</b>	<b>показатель интервала согласно диалогу установки параметров выносной надписи</b>
<b>LABEL_TEXT_WIDTH_FACT</b>	<b>показатель ширины символов согласно диалогу установки параметров выносной надписи</b>
<b>LABEL_TEXT_CHARSFACT</b>	<b>показатель межзнакового интервала согласно диалогу установки параметров выносной надписи</b>

## Параметры стен - доступны для дверей и окон

<b>WALL_ID</b>		использовать ID стены
<b>WALL_INTID</b>		внутренний ID стены
<b>WALL_INTGUID</b>		внутренний GUID стены, внутренний GUID, создаваемый программой <i>(не контролируется пользователем)</i>
<b>WALL_RESOL</b>	J~	3D-аппроксимация криволинейной стены <i>действует только в 3D</i>
<b>WALL_THICKNESS</b>	C_	толщина стены <i>в случае наклонных стен толщина стены определяется по оси открывания (ось z локальной системы координат)</i>
<b>WALL_START_THICKNESS</b>		толщина стены в начале
<b>WALL_END_THICKNESS</b>		толщина стены в конце
<b>WALL_INCL</b>		наклон поверхностей стен <i>угол между двумя наклонными поверхностями стен; 0 для обычных прямолинейных стен</i>
<b>WALL_HEIGHT</b>	D_	высота стены
<b>WALL_MAT_A</b>	G_	покрытие стены со стороны, противоположной открыванию
<b>WALL_MAT_B</b>	H_	покрытие стены со стороны, открывания <i>значение параметра может меняться от проема к проему, размещаемых в одной той же стене</i>
<b>WALL_MAT_EDGE</b>	I_	покрытие торца стены
<b>WALL_LINETYPE</b>		тип линии стены <i>применяется только к контурам стены в окне плана этажа</i>
<b>WALL_FILL</b>	A~	тип штриховки стены <i>индекс штриховки - в случае многослойных конструкций - это значение индекса первого слоя</i>
<b>WALL_FILL_PEN</b>	F_	перо штриховки стены
<b>WALL_COMPS_NAME</b>		многослойная конструкция стены <i>имя многослойной конструкции, изменяется от 1 и до 8, 0 - если применяется одна штриховка</i>
<b>WALL_SKINS_NUMBER</b>		количество слоев в многослойной или сложной стене <i>принимает значения от 1 и до 127, 0 - если применяется одна штриховка</i>

<b>WALL_SKINS_PARAMS</b>	<b>параметры слоев многослойных или сложных стен</b>
<i>двумерный массив, содержащий произвольное количество строк и 14 следующих столбцов: : образец штриховки, толщина, (старое перо контура), перо штриховки, перо фона штриховки, статус несущего слоя, перо верхней линии, тип верхней линии, перо нижней линии, тип нижней линии, перо конечной поверхности, ориентация штриховки, тип слоя, тип линии конечной поверхности.</i>	
<i>статус несущего слоя: 0 - не является составной частью, 1 - является составной частью, 3 - последний слой среди несущих слоев; ориентация штриховки: 0 - глобальная, 1 - локальная; тип слоя: 0 - сечение, 1 - ниже плоскости сечения, 2 - выше плоскости сечения (типы всех слоев для простых стен - 0). Для окон/дверей, расположенных в сложной стене, на плане этажа эта переменная содержит данные всех слоев в сечении, для концов стены на плане этажа эта переменная содержит данные всех слоев.</i>	
<i>Для окон/дверей и концов стен в 3D-окне эта переменная содержит данные слоев, которые действительно отсекаются окном/дверью или концом стены.</i>	
<b>WALL_SECT_PEN</b>	<b>E_ перо контура секущей поверхности стены</b>
<i>применяем к контурам секущих поверхностей стен как в окне плана этажа, так и в окнах разрезов/фасадов</i>	
<b>WALL_VIEW_PEN</b>	<b>перо контуров пространственного представления стены</b>
<i>применяется ко всем ребрам в 3D-окне и к видимым ребрам в окне разреза/фасада</i>	
<b>WALL_FBGD_PEN</b>	<b>перо фона штриховки стены</b>
<b>WALL_DIRECTION</b>	<b>ориентация стены</b>
<i>для прямолинейных стен: ориентация линии привязки, для криволинейных стен: ориентация хорды дуги</i>	
<b>WALL_POSITION</b>	<b>абсолютные координаты стены</b>
<i>положение начальной точки стены относительно начала системы координат проекта</i>	



## Параметры стен - доступны только в смете и выносных надписях

WALL_LENGTH_A	длина стены со стороны линии привязки.
WALL_LENGTH_B	длина стены со стороны, противоположной линии привязки
WALL_CENTER_LENGTH	длина стены в центре
WALL_AREA	площадь поверхности стены
WALL_PERIMETER	периметр стены
WALL_SURFACE_A	поверхность стены со стороны линии привязки
WALL_SURFACE_B	поверхность стены со стороны, противоположной линии привязки
WALL_GROSS_SURFACE_A	общая поверхность стены со стороны линии привязки
WALL_GROSS_SURFACE_B	общая поверхность стены со стороны, противоположной линии привязки
WALL_EDGE_SURF	поверхность торца стены
WALL_VOLUME	объем стены
WALL_GROSS_VOLUME	общий объем стены
WALL_DOORS_NR	число дверных проемов в стене
WALL_WINDS_NR	число оконных проемов в стене
WALL_HOLES_NR	число пустых проемов
WALL_DOORS_SURF	поверхность, занимаемая дверными проемами в стене
WALL_WINDS_SURF	поверхность, занимаемая оконными проемами в стене
WALL_HOLES_SURF	поверхность, занимаемая пустыми проемами
WALL_HOLES_SURF_A	аналитическая поверхность проемов со стороны линии привязки
WALL_HOLES_VOLUME	аналитический объем проемов в стене
WALL_WINDS_WID	суммарная ширина окон в стене
WALL_DOORS_WID	суммарная ширина дверей в стене
WALL_COLUMNS_NR	число размещенных в стене колонн
WALL_MIN_HEIGHT	минимальная высота стены
WALL_MAX_HEIGHT	максимальная высота стены
WALL_SECT_PEN	перо контура секущей поверхности стены, применяемое как на плане этажа, так и в окнах разрезов/фасадов
WALL_VIEW_PEN	перо контуров стены, применяемое ко всем ребрам в 3D-окне и к ребрам контуров (ребра элемента ниже плоскости сечения) на плане этажа и в окнах разрезов/фасадов

## Параметры колонн - доступны только в смете и выносных надписях

<b>COLU_CORE</b>	<b>спецификации ядра/облицовки</b> <i>используется только для совместимости: действителен только в скрипте спецификаций CPS-файла (спецификации колонны)</i>
<b>COLU_HEIGHT</b>	<b>высота колонны</b>
<b>COLU_MIN_HEIGHT</b>	<b>минимальная высота колонны</b>
<b>COLU_MAX_HEIGHT</b>	<b>максимальная высота колонны</b>
<b>COLU_VENEER_WIDTH</b>	<b>толщина облицовки колонны</b>
<b>COLU_CORE_X</b>	<b>ширина ядра</b>
<b>COLU_CORE_Y</b>	<b>толщина ядра</b>
<b>COLU_DIM1</b>	<b>1-й размер колонны</b>
<b>COLU_DIM2</b>	<b>2-й размер колонны</b>
<b>COLU_MAT</b>	<b>покрытие колонны</b>
<b>Примечание:</b> для встраиваемых колонн покрытие прилегающих стен будет замещать покрытие колонны	
<b>COLU_LINETYPE</b>	<b>тип линии для колонны</b> <i>применяется к контурам только в окне плана этажа</i>
<b>COLU_CORE_FILL</b>	<b>штриховка ядра колонны</b>
<b>COLU_VENEER_FILL</b>	<b>штриховка облицовки колонны</b>
<b>COLU_SECT_PEN</b>	<b>пера контуров секущей поверхности колонны</b> <i>применяется к контурам секущей поверхности как на плане этажа, так и в окнах разрезов/фасадов</i>
<b>COLU_VIEW_PEN</b>	<b>перо контура в 3D-виде</b> <i>применяется ко всем ребрам в 3D-окне и к ребрам контуров (ребра, располагающиеся ниже плоскости сечения) на плане этажа и в окнах разрезов/фасадов</i>
<b>COLU_CORE_FILL_PEN</b>	<b>перо штриховки ядра колонны</b>
<b>COLU_CORE_FBGD_PEN</b>	<b>перо фона штриховки ядра колонны</b>
<b>COLU_VENEER_FILL_PEN</b>	<b>перо штриховки облицовки колонны</b>
<b>COLU_VENEER_FBGD_PEN</b>	<b>перо фона штриховки облицовки колонны</b>
<b>COLU_PERIMETER</b>	<b>периметр колонны</b>
<b>COLU_AREA</b>	<b>площадь поверхности колонны</b>
<b>COLU_VOLUME</b>	<b>объем колонны</b>
<b>COLU_GROSS_VOLUME</b>	<b>общий объем колонны</b>
<b>COLU_CORE_SURF</b>	<b>поверхность ядра колонны</b>
<b>COLU_CORE_GROSS_SURF</b>	<b>общая поверхность колонны</b>
<b>COLU_CORE_VOL</b>	<b>объем ядра колонны</b>

---

<b>COLU_CORE_GROSS_VOL</b>	<b>общий объем ядра колонны</b>
<b>COLU_VENEER_SURF</b>	<b>поверхность облицовки колонны</b>
<b>COLU_VENEER_GROSS_SURF</b>	<b>общая поверхность облицовки колонны</b>
<b>COLU_VENEER_VOL</b>	<b>объем облицовки колонны</b>
<b>COLU_VENEER_GROSS_VOL</b>	<b>общий объем облицовки колонны</b>
<b>COLU_CORE_TOP_SURF</b>	<b>поверхность колонны сверху</b>
<b>COLU_CORE_BOT_SURF</b>	<b>поверхность колонны снизу</b>
<b>COLU_VENEER_TOP_SURF</b>	<b>поверхность облицовки сверху</b>
<b>COLU_VENEER_BOT_SURF</b>	<b>поверхность облицовки снизу</b>
<b>COLU_CORE_GROSS_TOPBOT_SURF</b>	<b>общая поверхность ядра сверху и снизу</b>
<b>COLU_VENEER_GROSS_TOPBOT_SURF</b>	<b>общая поверхность облицовки сверху и снизу</b>

---

## Параметры балок - доступны только в смете и выносных надписях

BEAM_THICKNESS	толщина балки
BEAM_HEIGHT	высота балки
BEAM_REFLINE_OFFSET	смещение линии привязки относительно оси балки
BEAM_PRIORITY	номер индекса приоритета 3D-пересечения
BEAM_MAT_RIGHT	покрытие балки с правой стороны от линии привязки
BEAM_MAT_LEFT	покрытие балки с левой стороны от линии привязки
BEAM_MAT_TOP	покрытие балки сверху
BEAM_MAT_BOTTOM	покрытие балки снизу
BEAM_MAT_END	покрытие балки в обоих торцах
BEAM_OUTLINE_LINETYPE	тип линии контура балки
BEAM_AXES_LINETYPE	тип линии оси балки
BEAM_FILL	образец штриховки балки
BEAM_FILL_PEN	перо штриховки балки
BEAM_SECT_PEN	перо контуров секущей поверхности балки
BEAM_FBGD_PEN	перо фона штриховки балки
BEAM_DIRECTION	ориентация линии привязки балки
BEAM_POSITION	абсолютные координаты начальной точки оси балки
BEAM_LENGTH_RIGHT	длина балки справа от линии привязки
BEAM_LENGTH_LEFT	длина балки слева от линии привязки
BEAM_RIGHT_SURF	поверхность балки справа от линии привязки
BEAM_LEFT_SURF	поверхность балки слева от линии привязки
BEAM_TOP_SURF	поверхность балки сверху
BEAM_BOTTOM_SURF	поверхность балки снизу
BEAM_END_SURF	поверхность балки в торцах
BEAM_VOLUME	объем балки
BEAM_HOLES_NR	количество отверстий в балке
BEAM_HOLES_SURF	общая поверхность отверстий в балке
BEAM_HOLE_EDGE_SURF	общая поверхность торцов отверстий в балке
BEAM_HOLES_VOLUME	общий объем отверстий в балке
BEAM_SECT_PEN	перо контуров секущей поверхности балки

## Параметры перекрытий - доступны только в смете и выносных надписях

<b>SLAB_THICKNESS</b>	толщина перекрытия
<b>SLAB_MAT_TOP</b>	покрытие верхней поверхности перекрытия
<b>SLAB_MAT_EDGE</b>	покрытие торцов перекрытия
<b>SLAB_MAT_BOTT</b>	покрытие нижней поверхности перекрытия
<b>SLAB_LINETYPE</b>	тип линии перекрытия
<b>SLAB_FILL</b>	образец штриховки перекрытия
	<i>индекс образца штриховки - в случае многослойных конструкций значение индекса отрицательное</i>
<b>SLAB_FILL_PEN</b>	перо штриховки перекрытия
<b>SLAB_FBGD_PEN</b>	перо фона штриховки перекрытия
<b>SLAB_COMPS_NAME</b>	многослойная конструкция перекрытия
	<i>наименование многослойной конструкции</i>
<b>SLAB_SKINS_NUMBER</b>	число слоев в многослойной конструкции
	<i>принимает значение от 1 до 8, при использовании единственного образца штриховки равен 0</i>
<b>SLAB_SKINS_PARAMS</b>	параметры слоев многослойного перекрытия
	<i>двумерный массив из 8 строк и 12 столбцов со следующими значениями: образец штриховки, толщина, (рашпилье - перо контура), перо штриховки, перо фона штриховки, статус несущего слоя, перо верхней линии, перо нижней линии внизу, тип верхней линии, тип нижней линии, перо конечной поверхности, ориентация штриховки; статус несущего слоя: 0 - не часть, 1 - часть, 3 - последний слой среди несущих слоев; ориентация штриховки: 0 - глобальная, 1 - локальная</i>
<b>SLAB_SECT_PEN</b>	перо контуров перекрытий в разрезе
	<i>применяется к контурам секущих поверхностей как в окне плана этажа, так и в окнах разрезов/фасадов</i>
<b>SLAB_VIEW_PEN</b>	перо перекрытия
	<i>применяется ко всем ребрам перекрытий в 3D-окне и к видимым ребрам в окнах разрезов/фасадов</i>
<b>SLAB_TOP_SURF</b>	верхняя поверхность перекрытия
<b>SLAB_GROSS_TOP_SURF</b>	общая верхняя поверхность перекрытия
<b>SLAB_BOT_SURF</b>	нижняя поверхность перекрытия
<b>SLAB_GROSS_BOT_SURF</b>	общая нижняя поверхность перекрытия
<b>SLAB_EDGE_SURF</b>	поверхность торцов перекрытия
<b>SLAB_GROSS_EDGE_SURF</b>	общая поверхность торцов перекрытия
<b>SLAB_PERIMETER</b>	периметр перекрытия
<b>SLAB_VOLUME</b>	объем перекрытия
<b>SLAB_GROSS_VOLUME</b>	общий объем перекрытия
<b>SLAB_SEGMENTS_NR</b>	количество сегментов перекрытия

<b>SLAB_HOLES_NR</b>	<b>число отверстий в перекрытии</b>
<b>SLAB_HOLES_AREA</b>	<b>площадь, занимаемая отверстиями в перекрытии</b>
<b>SLAB_HOLES_PRM</b>	<b>периметр отверстий, размещенных в перекрытии</b>

## Параметры крыш - доступны только в смете и выносных надписях

<b>ROOF_THICKNESS</b>	<b>толщина крыши</b>
<b>ROOF_ANGLE</b>	<b>наклон ската крыши</b>
<b>ROOF_MAT_TOP</b>	<b>покрытие верхней поверхности крыши</b>
<b>ROOF_MAT_EDGE</b>	<b>покрытие торцов крыши</b>
<b>ROOF_MAT_BOTT</b>	<b>покрытие нижней поверхности крыши</b>
<b>ROOF_LINETYPE</b>	<b>тип линии крыши</b>
<i>применяется к контурам только в окне плана этажа</i>	
<b>ROOF_FILL</b>	<b>штриховка крыши</b>
<i>индекс образца штриховки - в случае многослойных конструкций значение индекса отрицательно</i>	
<b>ROOF_FILL_PEN</b>	<b>перо штриховки крыши</b>
<b>ROOF_FBGD_PEN</b>	<b>перо фона штриховки крыши</b>
<b>ROOF_COMPS_NAME</b>	<b>многослойная конструкция крыши</b>
<i>наименование многослойной конструкции</i>	
<b>ROOF_SKINS_NUMBER</b>	<b>количество слоев многослойной крыши</b>
<i>принимает значения от 1 до 8; равен 0 в случае использования единственного образца штриховки</i>	
<b>ROOF_SKINS_PARAMS</b>	<b>параметры слоев многослойной конструкции</b>
<i>двумерный массив из 8 строк и 12 столбцов со следующими значениями: образец штриховки, толщина, (старое перо контура), перо штриховки, перо фона штриховки, статус несущего слоя, перо верхней линии, тип верхней линии, перо нижней линии, тип нижней линии, перо конечной поверхности, ориентация штриховки, статус несущего слоя: 0 - не часть, 1 - часть, 3 - последний слой среди несущих слоев; ориентация штриховки: 0 - глобальная, 1 - локальная;</i>	
<b>ROOF_SECT_PEN</b>	<b>перо контура секущих поверхностей крыши</b>
<i>применяется к контурам секущих поверхностей крыши как в окне плана этажа, так и в окнах разрезов/фасадов.</i>	
<b>ROOF_VIEW_PEN</b>	<b>перо крыши в 3D-виде</b>
<i>применяется ко всем ребрам в 3D-окне и к ребрам контуров (ребра, располагающиеся ниже плоскости сечения) на плане этажа и в окнах разрезов/фасадов</i>	
<b>ROOF_BOTTOM_SURF</b>	<b>нижняя поверхность крыши</b>
<b>ROOF_GROSS_BOTTOM_SURF</b>	<b>общая нижняя поверхность крыши</b>
<b>ROOF_TOP_SURF</b>	<b>верхняя поверхность крыши</b>
<b>ROOF_GROSS_TOP_SURF</b>	<b>общая верхняя поверхность крыши</b>

<b>ROOF_EDGE_SURF</b>	торцевая поверхность крыши
<b>ROOF_GROSS_EDGE_SURF</b>	общая торцевая поверхность крыши
<b>ROOF_PERIMETER</b>	периметр крыши
<b>ROOF_VOLUME</b>	объем крыши
<b>ROOF_GROSS_VOLUME</b>	общий объем крыши
<b>ROOF_SEGMENTS_NR</b>	число сегментов крыши
<b>ROOF_HOLES_NR</b>	число отверстий в крыше
<b>ROOF_HOLES_AREA</b>	площадь, занимаемая отверстиями в крыше
<b>ROOF_HOLES_PRM</b>	периметр отверстий в крыше

### **Параметры заштрихованных областей - доступны только в смете и выносных надписях**

<b>FILL_LINETYPE</b>	тип линии заштрихованной области
<b>FILL_FILL</b>	образец штриховки заштрихованной области
<b>FILL_FILL_PEN</b>	перо образца штриховки заштрихованной области
<b>FILL_PEN</b>	перо штриховки заштрихованной области
<b>FILL_FBGD_PEN</b>	перо фона штриховки заштрихованной области
<b>FILL_SURF</b>	площадь заштрихованной области
<b>FILL_PERIMETER</b>	периметр заштрихованной области
<b>FILL_SEGMENT_NR</b>	число сегментов заштрихованной области
<b>FILL_HOLES_NR</b>	число отверстий внутри заштрихованной области
<b>FILL_HOLES_PRM</b>	периметр отверстий заштрихованной области
<b>FILL_HOLES_AREA</b>	площадь, занимаемая отверстиями в заштрихованной области

## Параметры 3D-сетки - доступны только в смете и выносных надписях

<b>MESH_TYPE</b>	<b>тип 3D-сетки</b> <i>1 - тело, 2 - верхняя и боковая поверхности, 3 - только верхняя поверхность</i>
<b>MESH_BASE_OFFSET</b>	<b>смещение нижней поверхности от базового уровня</b>
<b>MESH_USEREDGE_PEN</b>	<b>перо ребер 3D-сетки, определяемых пользователем</b>
<b>MESH_TRIEDGE_PEN</b>	<b>перо триангуляционных ребер 3D-сетки</b>
<b>MESH_SECT_PEN</b>	<b>перо контуров 3D-сетки в разрезе</b> <i>применяется к контурам поверхностей сечения сеток в окне плана этажа и в окнах разрезов/фасадов</i>
<b>MESH_VIEW_PEN</b>	<b>перо контуров 3D-вида</b> <i>применяется ко всем контурам в 3D-окне и к ребрам в окнах разрезов/фасадов</i>
<b>MESH_MAT_TOP</b>	<b>покрытие верхней поверхности 3D-сетки</b>
<b>MESH_MAT_EDGE</b>	<b>покрытие боковой поверхности 3D-сетки</b>
<b>MESH_MAT_BOTT</b>	<b>покрытие нижней поверхности 3D-сетки</b>
<b>MESH_LINETYPE</b>	<b>тип линии 3D-сетки</b> <i>применяется к контурам только в окне плана этажа</i>
<b>MESH_FILL</b>	<b>образец штриховки 3D-сетки</b>
<b>MESH_FILL_PEN</b>	<b>перо штриховки 3D-сетки</b>
<b>MESH_FBGD_PEN</b>	<b>перо фона штриховки 3D-сетки</b>
<b>MESH_BOTTOM_SURF</b>	<b>нижняя поверхность 3D-сетки</b>
<b>MESH_TOP_SURF</b>	<b>верхняя поверхность 3D-сетки</b>
<b>MESH_EDGE_SURF</b>	<b>боковая поверхность 3D-сетки</b>
<b>MESH_PERIMETER</b>	<b>периметр 3D-сетки</b>
<b>MESH_VOLUME</b>	<b>объем 3D-сетки</b>
<b>MESH_SEGMENTS_NR</b>	<b>число сегментов 3D-сетки</b>
<b>MESH_HOLES_NR</b>	<b>количество отверстий в 3D-сетке</b>
<b>MESH_HOLES_AREA</b>	<b>площадь, занимаемая отверстиями в 3D-сетке</b>
<b>MESH_HOLES_PRM</b>	<b>периметр отверстий в 3D-сетке</b>



## Глобальные переменные свободные для использования

GLOB_USER_1	S_	
GLOB_USER_2	T_	
GLOB_USER_3	U_	
GLOB_USER_4	V_	
GLOB_USER_5	W_	
GLOB_USER_6	X_	
GLOB_USER_7	Y_	
GLOB_USER_8	Z_	
GLOB_USER_9	G~	
GLOB_USER_10	I~	свободные переменны 1 - 10 по умолчанию инициализируются как числовые
GLOB_USER_11		
GLOB_USER_12		
GLOB_USER_13		
GLOB_USER_14		
GLOB_USER_15		
GLOB_USER_16		
GLOB_USER_17		
GLOB_USER_18		
GLOB_USER_19		
GLOB_USER_20		свободные переменные 11 - 20 по умолчанию инициализируются как строковые

*Пример, иллюстрирующий использование глобальных переменных GLOB\_WORLD\_ORIGO\_... :*

```
GLOB_WORLD_ORIGO_... globals:
ADD2 -GLOB_WORLD_ORIGO_OFFSET_X - SYMB_POS_X, -GLOB_WORLD_ORIGO_OFFSET_X -SYMB_POS_Y
LINE2 -0.1, 0.0, 0.1, 0.0
LINE2 0.0, -0.1, 0.0, 0.1
HOTSPOT2 0.0, 0.0, 1
TEXT2 0, 0, "( 0.00 ; 0.00 )"
TEXT2 0, 0.5, "World Origo"
DEL TOP
if ABS(GLOB_WORLD_ORIGO_OFFSET_X) > 0.01 OR ABS(GLOB_WORLD_ORIGO_OFFSET_Y) > 0.01 THEN
  ADD2 - SYMB_POS_X, - SYMB_POS_Y
  LINE2 -0.1, 0.0, 0.1, 0.0
  LINE2 0.0, -0.1, 0.0, 0.1
  HOTSPOT2 0.0, 0.0, 2
  TEXT2 0, 0, "(" + STR (GLOB_WORLD_ORIGO_OFFSET_X, 9, 4) + "; " + STR
(GLOB_WORLD_ORIGO_OFFSET_Y, 9, 4) + " )"
  TEXT2 0, 0.5, "Virtual Origo"
  DEL TOP
ENDIF
if ABS(GLOB_WORLD_ORIGO_OFFSET_X + SYMB_POS_X) > 0.01 OR ABS(GLOB_WORLD_ORIGO_OFFSET_Y +
SYMB_POS_Y) > 0.01 THEN
  LINE2 -0.1, 0.0, 0.1, 0.0
  LINE2 0.0, -0.1, 0.0, 0.1
  HOTSPOT2 0.0, 0.0, 3
TEXT2 0, 0, "(" + STR (GLOB_WORLD_ORIGO_OFFSET_X + SYMB_POS_X, 9, 4) + "; " + STR
(GLOB_WORLD_ORIGO_OFFSET_Y + SYMB_POS_Y, 9, 4) + " )"
TEXT2 0, 0.5, "Object Placement"
ENDIF
```

## Старые глобальные переменные

Можно использовать имена старых глобальных переменных; однако, рекомендуется использовать новые имена. Каждой старой глобальной переменной соответствует новое длинное имя.

A_	GLOB_SCALE
B_	GLOB_HISTORY_ELEV
C_	WALL_THICKNESS
D_	WALL_HEIGHT
E_	WALL_SECT_PEN
F_	WALL_FILL_PEN
G_	WALL_MAT_A
H_	WALL_MAT_B
I_	WALL_MAT_EDGE
J_	GLOB_ELEVATION
K_	WIDO_SILL
L_	SYMB_VIEW_PEN
M_	SYMB_MAT
N_	GLOB_FRAME_NR
O_	GLOB_FIRST_FRAME
P_	GLOB_LAST_FRAME
Q_	GLOB_HISTORY_HEIGHT
R_	WIDO_ORIG_DIST
S_	GLOB_USER_1
T_	GLOB_USER_2
U_	GLOB_USER_3
V_	GLOB_USER_4
W_	GLOB_USER_5
X_	GLOB_USER_6
Y_	GLOB_USER_7
Z_	GLOB_USER_8
A~	WALL_FILL
B~	WIDO_RIGHT_JAMB
C~	WIDO_THRES_DEPTH
D~	WIDO_HEAD_DEPTH
E~	WIDO_REVEAL_SIDE

F~ WIDO\_FRAME\_THICKNESS  
G~ GLOB\_USER\_9  
H~ WIDO\_POSITION  
I~ GLOB\_USER\_10  
J~ WALL\_RESOL  
K~ GLOB\_EYEPOS\_X  
L~ GLOB\_EYEPOS\_Y  
M~ GLOB\_EYEPOS\_Z  
N~ GLOB\_TARGPOS\_X  
O~ GLOB\_TARGPOS\_Y  
P~ GLOB\_TARGPOS\_Z  
Q~ GLOB\_CSTORY\_ELEV  
R~ GLOB\_CSTORY\_HEIGHT  
S~ GLOB\_CH\_STORY\_DIST  
T~ GLOB\_SCRIPT\_TYPE  
U~ GLOB\_NORTH\_DIR  
V~ SYMB\_MIRRORED  
W~ SYMB\_ROTANGLE  
X~ SYMB\_POS\_X  
Y~ SYMB\_POS\_Y  
Z~ SYMB\_POS\_Z

## ЗАПРОСЫ

### REQ

**REQ** (parameter\_string)

Данная функция запрашивает текущее состояние программы. Ее параметр является строкой символов. GDL интерпретирует ответ числовым значением. Если строка-параметр неверная, то возвращаемое значение отрицательное..

*Допустимыми строками-параметрами являются:*

**"GDL\_version"**

Запрашивает номер версии компилятора/интерпретатора GDL.

**Внимание:** Это не то же самое, что версия ArchiCAD.

**"Program"**

Запрашивает код программы (например, 1: ArchiCAD).

**"Serial\_number"**

Запрашивает серийный номер электронного ключа защиты.

**"Model\_size"**

Запрашивает размер текущей структуры 3D-данных в байтах.

**"Red\_of\_material name"**

**"Green\_of\_material name"**

**"Blue\_of\_material name"**

Запрашивает значение соответствующей компоненты цвета (красный, зеленый, синий) указанного покрытия. Результат в интервале между 0 и 1.

**"Red\_of\_pen index"**

**"Green\_of\_pen index"**

**"Blue\_of\_pen index"**

Запрашивает значение соответствующей компоненты цвета (красный, зеленый, синий) указанного пера. Результат в интервале между 0 и 1.

**"Pen\_of\_RGB r g b"**

Запрашивает значение индекса пера, которое ближе всего подходит к указанному цвету. Параметры r, g, b содержат значения в интервале между 0 и 1.

## REQUEST

**REQUEST** (question\_name, name | index, variable1 [, variable2,...])

Первый параметр функции представляет текст вопроса, тогда как второй параметр представляет объект вопроса (если он существует) и может быть либо числового, либо строкового типа (например, вопросом может быть "Rgb\_of\_material", а его объектом - имя покрытия, или вопрос - "Rgb\_of\_pen", а объект вопроса - индекс пера). Другие параметры - это имена переменных, в которых хранятся возвращаемые значения (ответы). В качестве значения функции возвращается количество ответов (в случае плохо сформулированного вопроса или несуществующего имени значение функции будет равно 0).

**REQUEST** ("Name\_of\_program", "", program\_name)

Возвращает в заданную переменную имя программы, например, "ArchiCAD", и т.д.

*Пример: Печать имени программы*

```
n=REQUEST("Name_of_program", "", program_name)
PRINT program_name
```

**REQUEST** ("Name\_of\_macro", "", my\_name)

**REQUEST** ("Name\_of\_main", "", main\_name)

После вызова этих функций переменная my\_name будет содержать имя макроса, а main\_name - имя главного макроса (если он не существует, то пустую строку).

**REQUEST** ("ID\_of\_main", "", id\_string)

Для библиотечных элементов размещенных на плане этажа возвращает в переменную id\_string идентификатор, определенный в диалоговом окне установки параметров соответствующих инструментов (в противном случае возвращает пустую строку).

**REQUEST** ("Name\_of\_plan", "", name)

Возвращает в переменную name наименование текущего проекта.

**REQUEST** ("Story", "", index, story\_name)

Возвращает в переменные index и story\_name соответственно индекс и название текущего этажа.

**REQUEST** ("Home\_story", "", index, story\_name)

Возвращает в переменные index и story\_name variables соответственно индекс и название собственного этажа.

**REQUEST** ("Story\_info", expr, nStories,  
index1, name1, elev1, height1 [,  
index2, name2, ...])

Возвращает информацию об этаже в указанные переменные: количество этажей, индекс этажа, возвышение, высота. Если expr является числовым выражением, то это обозначает индекс этажа; при этом возвращается количество этажей и информация о запрашиваемом этаже. Если expr является строковым выражением, то это означает, что запрашивается информация обо всех этажах. Возвращаемое значение функции - количество успешно найденных значений.

*Пример:*

```
DIM t[]
  n = REQUEST ("STORY_INFO", "", nr, t)
  FOR i = 1 TO nr
    nr = STR ("%0m", t [4 * (i - 1) + 1])
    name = t [4 * (i - 1) + 2]
    elevation = STR ("%m", t [4 * (i - 1) + 3])
    height = STR ("%m", t [4 * (i - 1) + 4])
    TEXT2 0, -i, nr + "," + name + "," + elevation + "," + height
  NEXT i
```

**REQUEST ("Internal\_id", "", id)**

Возвращает в переменную id внутренний id библиотечного элемента.

**REQUEST ("Linear\_dimension", "",  
format\_string)**

**REQUEST ("Angular\_dimension", "",  
format\_string)**

**REQUEST ("Angular\_length\_dimension", ""  
format\_string)**

**REQUEST ("Radial\_dimension", "",  
format\_string)**

**REQUEST ("Level\_dimension", "",  
format\_string)**

**REQUEST ("Elevation\_dimension", "",  
format\_string)**

**REQUEST ("Window\_door\_dimension", "",  
format\_string)**

**REQUEST ("Sill\_height\_dimension", "",  
format\_string)**

**REQUEST ("Area\_dimension", ""  
format\_string)**

**REQUEST ("Calculation\_length\_unit", "",  
format\_string)**

**REQUEST ("Calculation\_area\_unit", "",  
format\_string)**

```
REQUEST ("Calculation_volume_unit", "",
        format_string)
```

```
REQUEST ("Calculation_angle_unit", "",
        format_string)
```

Эти запросы позволяют получить множество форматов единиц измерения, установленных в диалоговом окне ArchiCAD *Параметры/Рабочая среда проекта/Размерные числа*. Эти запросы возвращают формат, который может использоваться в качестве первого параметра в функции STR().

*Пример:*

```
format = ""
num = 60.55
REQUEST ("Angular_dimension", "", format)
! "%.2dd"
```

```
TEXT2 0, 0, STR (format, num)!60.55
```

```
REQUEST ("Clean_intersections", "", state)
```

Возвращает состояние команды *Скрывать сопряжение стен и балок* из меню *Вид/Параметры вывода на экран* (1, когда команда отмечена, 0 в противном случае).

```
REQUEST ("Zone_category", "", name, code)
```

Для зон возвращает имя и код текущей категории зоны.

```
REQUEST ("Zone_relations", "", category_name, code, name, number [ , category_name2,
        code2, name2, number2])
```

Возвращает в заданные переменные, соответственно, наименование категории зоны, код категории зоны, наименование и номер зоны, в которой размещен библиотечный элемент, вызывающий данную функцию. Для дверей и окон может существовать максимум две таких зоны. Результирующее значение запроса представляет число выявленных зон (если библиотечный элемент не попал ни в одну из зон, то 0).

```
REQUEST ("Zone_colus_area", "", area)
```

Возвращает в переменную area общую площадь, занимаемую колоннами, размещенными в текущей зоне. Действует только для паспортов зон.

```
REQUEST ("Custom_auto_label", "", name)
```

Возвращает в переменную name наименование специальной автоматически проставленной надписи библиотечного элемента или, если такой не существует, пустую строку.

```
REQUEST ("Rgb_of_material", name, r, g, b)
```

```
REQUEST ("Rgb_of_pen", penindex, r, g, b)
```

```
REQUEST ("Pen_of_RGB", "r g b", penindex)
```

Подобно функции REQ() (только одно обращение) возвращает в определенные переменные r, g, b значения компонент цвета (красный, зеленый, синий) покрытия и пера, или индекс пера, соответствующий заданным значениям компоненты цвета (красный, зеленый, синий).



```
REQUEST ("Height_of_style", name,
         height [, descent, leading])
```

Возвращает в заданную переменную высоту символов указанного стиля. Высота измеряется в миллиметрах (высота в метрах равна  $height / 1000 * GLOB\_SCALE$ ); descent - снижение (расстояние в миллиметрах от базовой линии текста до нижней линии) и leading - междустрочный интервал (расстояние в миллиметрах между нижней линией и верхней линией).

```
REQUEST ("Style_info", name, fontname [, size, anchor, face_or_slant])
```

Возвращает информацию в заданных переменных относительно ранее определенного стиля (см. параметры стиля в *"DEFINE STYLE"* на *стр. 176*). Может быть полезен в макросах для получения информации о стиле, определенном в основном скрипте.

```
REQUEST ("Name_of_material", index, name)
```

Возвращает в переменную name название покрытия по значению его индекса index.

```
REQUEST ("Name_of_fill", index, name)
```

Возвращает в переменную name наименование штриховки по значению ее индекса index.

```
REQUEST ("Name_of_line_type", index, name)
```

Возвращает в переменную name наименование типа линии по значению ее индекса index.

```
REQUEST ("Name_of_style", index, name)
```

Возвращает в переменную name название стиля по значению его индекса index.

Если  $index < 0$ , то обращение производится к покрытию, образцу штриховки, типу линии или стилю, определенному в скрипте GDL или в файле MASTER\_GDL. Вызов запроса с  $index=0$  возвращает в переменную name наименование покрытия или типа линии, установленных по умолчанию (Для штриховки и стиля возвращается пустая строка).

Возвращаемое значение запроса - успешность выполнения функции (1 - требуемое значение получено, 0 - ошибка в значении index).

```
REQUEST ("WINDOW_DOOR_SHOW_DIM", "", show)
```

До версии 9.0 функция возвращала значение 1 в переменной show, если *Параметры/Параметры вывода на экран/Двери и окна* установлен в *"Показать с размерами"*, 0 - в противном случае. Начиная с версии 9.0 параметры вывода на экран разделены для окон и дверей, поэтому для поддержания совместимости ArchiCAD проверяет, используется ли запрос для окна (или маркера окна) или для двери (или маркера двери) и автоматически возвращает соответствующее значение параметра вывода на экран. В других случаях (символ, источник света, выносная надпись) возвращается значение этого параметра для окна. Может использоваться, чтобы прятать/показывать размеры согласно текущим установкам вывода на экран.

Начиная с версии 9.0 появились два отдельных запроса: "window\_show\_dim" и "door\_show\_dim".

```
REQUEST ("window_show_dim", "", show)
```

Возвращает 1 в переменной show, если в диалоге команды *Документ/Установить модельный вид/Параметры модельного вида* для окон отмечен маркер *Показать на планах с маркерами*, 0 - в противном случае.

```
REQUEST ("door_show_dim", "", show)
```

Возвращает 1 в переменной show, если в диалоге команды *Документ/Установить модельный вид/Параметры модельного вида* для дверей отмечен маркер *Показать на планах с маркерами*, 0 - в противном случае.

**REQUEST** ("name\_of\_listed", " ", name)

Возвращает в переменной name имя библиотечного элемента, связанного с библиотечным элементом спецификации, содержащим этот запрос. Для элементов (стены, перекрытия и т.д.) name - это пустая строка.

**REQUEST** ("window\_door\_zone\_relev", " ", out\_direction)

Действует только для окон и дверей. Используйте этот запрос как дополнение к запросу "ZONE\_RELATIONS". Возвращает 1 в переменной out\_direction, если ориентация оконного/дверного проема направлена в сторону первой комнаты, определенной запросом "ZONE\_RELATIONS", 2 - если ориентация проема направлена в сторону второй комнаты. Также возвращается 2, если имеется только одна комната и ориентация направлена не в ее сторону.

**REQUEST** ("matching\_properties", type, name1, name2, ...)

Если type = 1, то в заданных переменных возвращаются имена индивидуально ассоциируемых библиотечных элементов спецификаций, в противном случае возвращаются имена библиотечных элементов спецификаций, специфицированных по критерию. Если используется в ассоциативной выносной надписи, то эта функция возвращает спецификации элемента, с которым эта выносная надпись ассоциирована.

**REQUEST** (extension\_name, parameter\_string, variable1, variable2, ...)

Если вопрос не совпадает ни с одним из выше перечисленных, то функция REQUEST () попытается проинтерпретировать его как специфическое имя расширения. Если это расширение находится в папке *Расширения ArchiCAD*, то оно будет использовано для получения как можно больше значений для как можно большего числа указанных в запросе имен переменных. Параметр parameter\_string интерпретируется расширением.

**REQUEST** ("Constr\_Fills\_display", "", optionVal)

Возвращает в заданной переменной значения параметра показа штриховки сечения, установленного в *Документ/Установить модельный вид/Параметры модельного вида* (предыдущая штриховка конструкций). Возможными значениями являются следующие.

- 1 - Показать только контур штриховки сечения (предыдущая - Пусто).
- 2 - Показать только контур штриховки сечения с линиями-разделителями слоев (Предыдущая - Нет штриховки).
- 4 - Образец штриховки сечения: Сплошная (предыдущая - Сплошная).
- 6 - Образец штриховки сечения: Согласно параметрам (предыдущая - Векторная штриховка).

Сама функция возвращает значение успешно найденных значений, 0 - если встретилась ошибка.

**REQUEST** ("Working\_length\_unit", "", format\_string)

**REQUEST** ("Working\_angle\_unit", "", format\_string)

С помощью этих запросов можно узнать форматы рабочих единиц, установленных в диалоге команды *Параметры > Рабочая среда проекта > Единицы измерения и уровни*. Эти запросы возвращают формат, который может использоваться в качестве первого параметра в функции STR(). Эти запросы оказываются действенными только когда производится интерпретация параметра или в скриптах интерфейса пользователя.

**IND** (MATERIAL, name\_string)  
**IND** (FILL, name\_string)  
**IND** (LINE\_TYPE, name\_string)  
**IND** (STYLE, name\_string)  
**IND** (TEXTURE, name\_string)

Возвращает текущий индекс реквизитов покрытия, образца штриховки, типа линии или стиля, соответственно. Основное использование получаемых значений заключается в том, чтобы передать их во внутренний макрос, который должен работать с тем же реквизитом, что и внешний макрос. Результат отрицательный для временного определения и положительный для глобального определения.

*См. также раздел “Реквизиты” в главе “Конфигурация” справки ArchiCAD, а также “Определение реквизитов” на стр. 162.*

**REQUEST** ("ASSOCLP\_PARVALUE", expr, name\_or\_index, type, flags, dim1, dim2, p\_values)

Возвращает информацию в заданной переменной относительно параметра библиотечного элемента, с которым связан библиотечный элемент, содержащий этот запрос. Может использоваться в объектах спецификаций, выносных надписях и в объектах маркеров.

Сама функция возвращает значение успешно найденных значений, 0 - если указанный параметр не существует или произошла ошибка.

expr: объект запроса, имя параметра ассоциированного библиотечного элемента или индекс выражения.

name\_or\_index: возвращает индекс или имя параметра в зависимости от типа предыдущего выражения (возвращает индекс, если предыдущее выражение - имя параметра; возвращает имя, если указан индекс)

type: тип параметра, возможные значения:

- 1: логический,
- 2: целое число,
- 3: действительное число,
- 4: строка,
- 5: длина,
- 6: угол,
- 7: линия,
- 8: покрытие,
- 9: штриховка,
- 10: цвет пера,
- 11: переключатель света,
- 12: цвет RGB,
- 13: интенсивность света
- 14: разделитель,
- 15: заголовок

flags:

flags = j1 + 2 \* j2 + 64 \* j7 + 128 \* j8,  
где j<sub>i</sub> может принимать значения 0 или 1.

j1 (1): является дочерним.

j2 (2): выделяется жирным стилем.

j7 (64): деактивирован.

j8 (128): скрыт.

dim1, dim2: возвращаются размеры параметра; оба 0 если параметр простой; dim1 > 0, dim2 = 0, если одномерный массив; оба > 0, если двумерный массив. dim1 - количество строк, dim2 - количество столбцов.

p\_values: возвращает значение параметра ала массив значений. Элементы массива возвращаются последовательно, строка за строкой, как одномерный массив, независимо от размеров переменной, указанной для их запоминания. Если переменная не является динамическим массивом, то будет запомнено столько элементов, сколько хватит места (для простой переменной - только один, первый, элемент). Если значения являются двумерным динамическим массивом, то все элементы запоминаются в первой строке.

**REQUEST** ("ASSOCLP\_NAME", "", name)

Возвращает в заданной переменной имя библиотечного элемента, ассоциируемого с выносной надписью или объектом маркера. Для элементов (стен, перекрытий и т.д.) это имя - пустая строка.

**REQUEST** ("ASSOCEL\_PROPERTIES ", parameter\_string, nr\_data, data)

Возвращает в заданных переменных собственные данные спецификации или спецификации элемента, с которым ассоциирован библиотечный элемент, содержащий этот запрос (в выносных надписях и ассоциативных объектах маркеров). Сама функция возвращает значение успешно найденных значений, 0 - если не были найдены данные спецификации или была обнаружена ошибка. Эта функция не работает в объектах спецификаций по составлению смет.

parameter\_string: последовательность ключевых слов, разделяемых запятыми, которые представляют запрашиваемые поля записей данных спецификаций. Записи будут упорядочены соответствующим образом. Возможные значения:

ISCOMP  
DBSETNAME  
KEYCODE  
KEYNAME  
CODE  
NAME  
FULLNAME  
QUANTITY  
TOTQUANTITY  
UNITCODE

UNITNAME  
 UNITFORMATSTR  
 PROPOBJNAME

**nr\_data**: возвращается количество элементов данных.

**data**: возвращаются данные спецификации; записи, содержащие поля, указанные в соответствующем параметре.

Значения возвращаются как одномерный массив, который содержит запрошенные поля записи, не зависимо от размеров переменной, указанной для их запоминания. Если переменная не является динамическим массивом, то будет запомнено столько элементов, сколько хватит места (для простой переменной - только один, первый, элемент). Если значения являются двумерным динамическим массивом, то все элементы запоминаются в первой строке.

*Пример:*

```
DIM DATA []
n = REQUEST ("ASSOCEL_PROPERTIES", "iscomp, code, name", nr, data)
  IF nr = 0 THEN
    TEXT2 0, 0, "No properties"
ELSE
  j = 0
  FOR i = 1 TO nr
    IF (i) MOD 3 = 0 THEN
      TEXT2 0, -j, DATA [i] ! name
      j = j + 1
    ENDIF
  NEXT i
ENDIF
REQUEST ("REFERENCE_LEVEL_DATA", "",
  name1, elev1, name2, elev2, name3, elev3)
```

Возвращает в заданных переменных имена и возвышения уровней привязки, заданные в диалоге команды *Параметры/Рабочая среда проекта/Единицы измерения и уровни*.

Сама функция возвращает количество успешно найденных значений; 0 - если была обнаружена ошибка.

```
REQUEST ("ANCESTRY_INFO", expr, name [,
  guid, parent_name1, parent_guid1,
  ...,
  parent_namen, parent_guidn)
```

Информация о "предке" библиотечного элемента.

Если *expr = 0*, то возвращает в заданных переменных имя и глобальный уникальный идентификатор библиотечного элемента, содержащего этот запрос. Дополнительно эта функция возвращает имена и

глобальные уникальные идентификаторы предков этого библиотечного элемента (parent\_namei, parent\_guidi). Если родительские шаблоны не загружены, то их имена становятся пустыми строками. Если *expr* = 1, то возвращается информация библиотечного элемента, замененного шаблоном, содержащим эту функцию. В этом случае, если шаблон фактически не заменяется, то никакие значения не возвращаются. Сама функция возвращает количество успешно найденных значений.

*Пример:*

```
DIM strings[]
n = REQUEST ("ANCESTRY_INFO", 1, name, guid, strings)
IF n > 2 THEN
  ! данные замененного библиотечного элемента
  TEXT2 0, -1, "замена: " + name + ' ' + guid
  ! родители
  l = -2
  FOR i = 1 to n - 2 STEP 2
    TEXT2 0, 1, strings [i]
    l = l - 1
  NEXT i
ENDIF
```

```
REQUEST ('TEXTBLOCK_INFO',
  textblock_name, width, height)
```

Возвращает в заданных переменных размер во направлениях x и y ранее определенного TEXTBLOCK. Размер в миллиметрах или метрах в модельном пространстве в зависимости от значения параметра *fixed\_height* в TEXTBLOCK (миллиметры, если 1, метры в модельном пространстве, если 0). Если параметр *width* был 0, то запрос возвращает вычисленную ширину и высоту, если параметр *width* был задан в определении TEXTBLOCK, то возвращается вычисленная высота, соответствующая этому параметру *width*.

```
REQUEST{2} ("Material_info", name_or_index,
  param_name, value_or_values)
```

```
REQUEST{2} ("Material_info", name_or_index,
  extra_param_name,
  value_or_values)
```

Возвращает информацию в заданных переменных относительно параметра (или дополнительного параметра, см. *“Дополнительные данные” на стр. 179*) указанного покрытия. Информация о цвете RGB возвращается в три отдельных переменных, информация о текстуре возвращается в следующие переменные : *file\_name*, *width*, *height*, *mask*, *rotation\_angle*, соответствующие определению текстуры. Вся остальная информация о параметрах возвращается в отдельных переменных. Возможные имена параметров покрытия, соответствующие параметрам определения покрытия:

```
gs_mat_surface_rgb (поверхность R, G, B [0.0..1.0])
gs_mat_ambient (коэффициент рассеивания [0.0..1.0])
```

gs\_mat\_diffuse (коэффициент диффузии [0.0..1.0])  
 gs\_mat\_specular (коэффициент зеркального отражения [0.0..1.0])  
 gs\_mat\_transparent (коэффициент прозрачности [0.0..1.0])  
 gs\_mat\_shining (фокусирование бликов[0.0..100.0])  
 gs\_mat\_transp\_att (анизотропный эффект [0.0..4.0])  
 gs\_mat\_specular\_rgb (цвет отражения R, G, B [0.0..1.0])  
 gs\_mat\_emission\_rgb (цвет люминесцентности R, G, B [0.0..1.0])  
 gs\_mat\_emission\_att (затухание люминесцентности [0.0..65.5])  
 gs\_mat\_fill\_ind (индекс штриховки)  
 gs\_mat\_fillcolor\_ind (индекс цвета штриховки)  
 gs\_mat\_texture (индекс текстуры)

*Пример:*

```

REQUEST{2} ("Material_info",
  "Brick-Face", "gs_mat_ambient", a)
REQUEST{2} ("Material_info", 1,
  "gs_mat_surface_rgb", r, g, b)
REQUEST{2} ("Material_info",
  "Brick-Face", "gs_mat_texture",
  file_name, w, h, mask, alpha)
REQUEST{2} ("Material_info",
  "My-Material", "my_extra_parameter", e)
REQUEST ("FONTNAMES_LIST", "", fontnames)
  
```

Возвращает в заданных переменных имена шрифтов, имеющихся на компьютере (вместе с кодами символов). Этот список (или любая часть этого списка) может использоваться в команде VALUES, чтобы создать всплывающее меню шрифтов. Сама функция возвращает значение успешно найденных значений, 0 - если встретилась ошибка.

*Пример:*

```

dim fontnames[]
REQUEST ("FONTNAMES_LIST", "", fontnames)
VALUES "f" fontnames, CUSTOM
  
```

Этот вид команды VALUES собирает имена шрифтов в один строковый параметр "f" простого типа для создания всплывающего меню. Переменная "fontnames" содержит возможные имена шрифтов (вместе с кодами символов), которые могут быть перечислены вручную или получены с помощью команды REQUEST ("FONTNAMES\_LIST", ...). Ключевое слово CUSTOM необходимо для правильного оперирования отсутствующими шрифтами на других платформах/компьютерах: если оно указано, то имя шрифта, выбранное на другом компьютере/платформе и отсутствующее в текущей окружающей среде, будет сохранено в параметре как специальное значение (в противном случае, согласно реализации команды VALUES, отсутствующее строковое значение всплывающего меню в установках параметра будет заменено на первое текущее строковое значение).

---

**REQUEST** ("HomeDB\_info", "", homeDBIntId, homeDBUserId, homeDBName, homeContext)

Возвращает в заданных переменных внутренний ID (целое число), ID пользователя и имя пользователя (строка) той базы данных, где размещен библиотечный элемент, содержащий этот запрос.

- при размещении на плане этажа: внутренний ID этажа, индекс как строка и имя, homeContext = **1**
- при размещении в разрезе: внутренний ID разреза, ссылочный ID и имя, homeContext = **2**
- при размещении в детали: внутренний ID детали, ссылочный ID и имя, homeContext = **3**
- при размещении в основном макете: внутренний ID макета, пустая строка и имя, homeContext = **4**
- при размещении в макете: внутренний ID макета, номер и имя, homeContext = **5**

Для выносных надписей возвращаемые данные относятся к элементу, на который эта выносная надпись ссылается. Вместе с GLOB\_INTID собранные данные могут использоваться для уникальной идентификации элементов в различных базах данных ArchiCAD файла плана этажа.

**REQUEST** ("floor\_plan\_option", ""storyViewpointType)

Возвращает тип взгляда этажа, установленный в параметрах модельного вида. 0 - "План этажа", 1 - "План потолка".

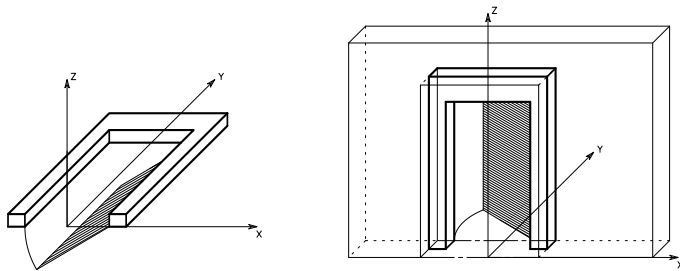


## ДВЕРИ И ОКНА

В этом разделе рассматриваются специальные возможности для создания библиотечных элементов окон и дверей.

### Общие положения

Сразу после размещения в стене дверного/оконного проема система координат этого библиотечного элемента, установленная по умолчанию, поворачивается таким образом, чтобы плоскость  $x$ - $y$  была расположена вертикально, а ось  $z$  горизонтально по отношению к стене. Начало координат размещается по центру нижней части проема, с внешней стороны стены. Такое расположение системы координат позволяет легко моделировать дверные/оконные проемы элементами в плоскости  $x$ - $y$ . См. рисунок ниже.



В силу специфики поведения этих библиотечных элементов, их 2D-символ создается из специальной встроенной проекции, которая недоступна пользователям (вид сбоку при взгляде сверху вниз под углом 90 градусов). Символ и 3D-фигура привязываются к началу координат окна/двери серединой ( $x$ ) нижней части ( $y$ ) рамки, ограничивающей фигуру, а привязки по оси  $z$  нет, чтобы дать возможность пользователю создавать окна и двери со смещением относительно данной оси в любом направлении.

С учетом выше изложенных правил, дадим несколько советов, которые помогут Вам правильно конструировать дверные/оконные проемы:

- Размещая дверь/окно на плане этажа, представьте себе, что Вы смотрите на нее с внутренней стороны той стены, в которую она/оно встраивается.
- Мысленно представьте себе, что нулевой уровень проекта совпадает с внешней поверхностью стены.
- Элементы, которые предстоит разместить в стене, например, оконная рама, должны быть выше нулевого уровня.
- Дверные полотна, открывающиеся наружу, должны быть ниже нулевого уровня.

## Создание библиотечных элементов дверей и окон

Имеются различные варианты создания дверей и окон, каждый из которых обладает специфическими проблемами.

Создание прямоугольной двери/окна в прямолинейной стене.

- 3D-варианты:
  - создание непрямоугольных окон/дверей в прямолинейных стенах;
  - создание прямоугольных окон/дверей в прямолинейных стенах;
  - создание непрямоугольных окон/дверей в криволинейных стенах.
- 2D-варианты:
  - Вырезание специальных проемов в стене
    - WALLHOLE2
  - Расширение многоугольника стены
    - WALLBLOCK2
    - WALLLINE2

## 3D-варианты

### Прямоугольные окна и двери в прямолинейных стенах

Это простейший и наиболее распространенный способ построения дверей и окон. В данном случае рекомендуется использовать простые команды GDL такие, как PRISM\_ и RECT.

Если Вы хотите подобрать покрытия поверхностей элементов двери/окна под покрытия стены, то нижняя поверхность элементов должна соответствовать покрытию внешней стороны стены, а верхняя - внутренней. В GDL-скрипте этого можно достичь с помощью глобальных переменных WALL\_MAT\_A, WALL\_MAT\_B и WALL\_MAT\_EDGE, предоставляющих информацию о покрытии стены, в которую встраивается дверь/окно. В 2D-скрипте можно воспользоваться глобальными переменными WALL\_SECT\_PEN, WALL\_FILL\_PEN и WALL\_FILL. Значения этих переменных задают номер пера для контура и штриховки стены и индекс образца штриховки стены на плане этажа, где размещается дверь или окно. Для многослойных стен следует использовать соответствующие глобальные переменные для многослойных конструкций.

См. *“Разное” на стр. 223* для получения подробной информации.

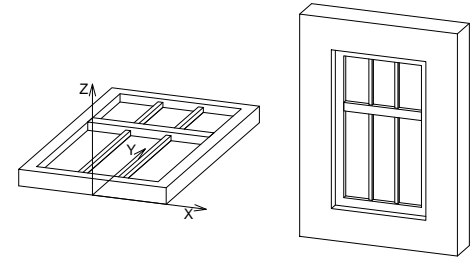
Библиотека ArchiCAD обладает достаточно обширным набором макросов для построения дверных и оконных проемов. Эти GDL-скрипты содержат общие элементы построения, используемые большинством дверей/окон библиотеки ArchiCAD. Эти макросы предназначены для создания наиболее часто используемых оконных рам, дверных коробок, стилей филленчатых полотен и многих других компонент окон и дверей. Откройте несколько произвольных библиотечных элементов окон или дверей и посмотрите, какие типы макросов вызываются в их скриптах и какие типы элементов эти макросы генерируют.

Пример:

```

A=0.9: B=1.5: C=0.1: D=0.08
E=0.08: F=0.9: G=0.03: H=3
PRISM 10,C,
  -A/2, 0, 15, A/2, 0, 15,
  A/2, B, 15, -A/2, B, 15,
  -A/2, 0, -1,
  -A/2+D, D, 15, A/2-D, D, 15,
  A/2-D, B-D, 15, -A/2+D, B-D, 15,
  -A/2+D, D, -1
ADD -A/2+D, F, 0
BRICK A-2*D, E, C
ADD -G/2, -F+D, C/2
GOSUB 1
ADDZ -G
GOSUB 1
DEL 2
MATERIAL "Стекло"
ADD0, -F+D, C/2
RECT A-2*D, F-D
ADDY F-D+E
RECT A-2*D, B-F-E-D
END
1: FOR I=1 TO H-1
ADDX (A-2*D)/3
BLOCK G, F-D, G
ADDY F+E-D
BLOCK G, B-F-D-E, G
DEL 1
NEXT I
DEL H-1
RETURN

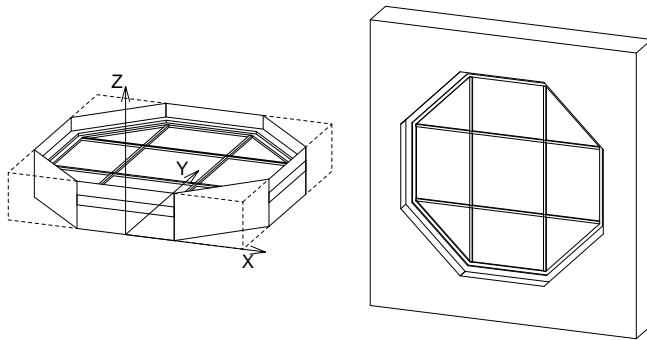
```



## Непрямоугольные двери и окна в прямолинейных стенах

Когда Вы работаете с окнами и дверями важно осознавать, что ArchiCAD всегда вырезает прямоугольное отверстие в стене, в которой размещается дверь или окно. Размеры отверстия определяются значениями параметров А и В библиотечного элемента двери/окна. Если же дверь/окно имеет прямоугольную форму, то оно не может полностью заполнить это прямоугольное отверстие, вырезанное в стене. Решением этой проблемы является использование команды WALLHOLE или WALLNICHE для определения многоугольной фигуры, вырезающей отверстие в стене, в котором будет размещаться окно или дверь. В этом случае возможны следующие две ситуации.

- 3D-скрипт должен содержать команды, восстанавливающие те части стены, которые оказались пустыми в результате устройства прямоугольного окна/двери. В этом случае, особое внимание следует уделить видимости ребер этих заполнений.



- С помощью команды WALLHOLE или WALLNICHE Вы можете определить многоугольную фигуру, которая будет использоваться в качестве секущей в том месте стены, где размещается дверной или оконный проем.

### WALLHOLE

```
WALLHOLE n, status,
         x1, y1, mask1,
         ...
         xn, yn, maskn
         [, x, y, z]
```

n: число вершин многоугольника.

status:

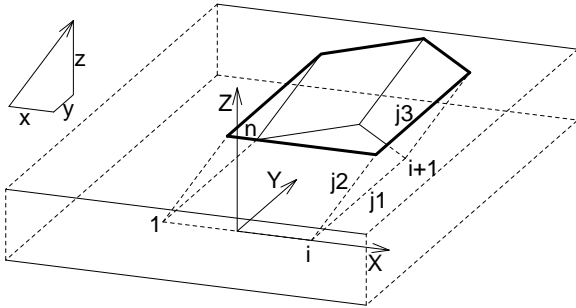
- 1: для создания многоугольников и ребер используются собственные атрибуты фигуры.
- 2: создаваемые секущие многоугольники рассматриваются как обычные многоугольники.

$x_i, y_i$ : координаты вершин многоугольника сечения.

maski: аналогично предложению CUTPOLYA:

$$\text{maski} = j1 + 2 * j2 + 4 * j3 + 64 * j7$$

x, y, z: факультативный вектор, задающий ориентацию сечения (по умолчанию ось z окна/двери).



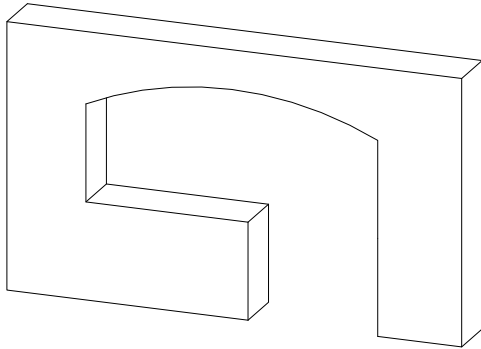
Эту команду можно использовать в 3D-скриптах дверей/окон для создания специальных отверстий в стенах, где они размещаются. В ходе 3D-генерации текущей стены производится интерпретация всех 3D-скриптов дверей и окон, имеющих в этой стене, без генерации модели, чтобы собрать все команды WALLHOLE. Если такие имеются, ArchiCAD построит сечение в текущей стене с помощью бесконечной многоугольной призмы в направлении, заданном в скрипте. Скрипт может содержать произвольное количество предложений WALLHOLE для любого окна/двери, таким образом, может быть вырезано несколько, даже пересекающихся, отверстий для одного и того же дверного/ оконного проема. Если 3D-скрипт окна/двери содержит хотя бы одно предложение WALLHOLE, то ArchiCAD для этого окна/двери не будет создавать прямоугольного отверстия.

**Примечание:** Для специальных проемов 3D-четверть не создается автоматически, ее необходимо определить в скрипте.

Определенный таким образом проем будет виден только в 3D-представлении, так как на плоскости выполнение предложения WALLHOLE не имеет какого-либо эффекта. Если необходимо 2D-представление может быть описано скриптом (при этом следует отключить *Изображение на плане*).

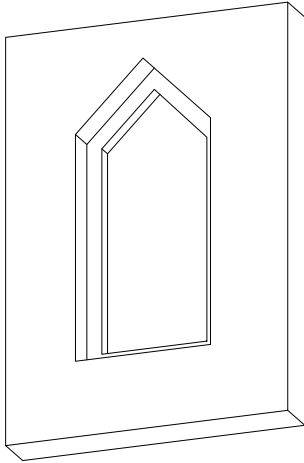
Рекомендуется в качестве секущих использовать выпуклые многоугольники; использование вогнутых многоугольников может дать необычный эффект при построении теней/реалистических изображений или привести к ошибкам при построении сечения. Вогнутые многоугольники могут быть получены комбинированием выпуклых многоугольников.

Примеры:



```

RESOL 72
L1=2.7: L2=1.2: H1=2.1: H2=0.3: H3=0.9
R= ((L1/2)^2+H2^2)/(2*H2)
A=ATN((L1/2)/(R-H2))
WALLHOLE 5,1,
  -L1/2,H3,15,
  L1/2,H3,15,
  L1/2,H1-H2,13,
  0,H1-R,915,
  0,2*A,4015
WALLHOLE 4,1,
  L1/2-L2,0,15,
  L1/2,0,15,
  L1/2,H3,15,
  L1/2-L2,H3,15
  
```



```
WALLHOLE 5,1,  
-0.45, 0, 15,  
0.45, 0, 15,  
0.45, 1.5, 15,  
0, 1.95, 15,  
-0.45, 1.5, 15
```

```
PRISM 12, 0.1,  
-0.45, 0, 15,  
0.45, 0, 15,  
0.45, 1.5, 15,  
0, 1.95, 15,  
-0.45, 1.5, 15,  
-0.45, 0, -1,  
-0.35, 0.1, 15,  
0.35, 0.1, 15,  
0.35, 1.45, 15,  
0, 1.80, 15,  
-0.35, 1.44, 15,  
-0.35, 0.1, -1
```

## WALLNICHE

```
WALLNICHE n, method, status,
  rx, ry, rz, d,
  x1, y1, mask1,
  ...
  xn, yn, maskn
```

Эта команда аналогична определению CUTFORM.

method: управляет формой тела сечения.

1: имеет форму призмы.

2: имеет форму пирамиды.

3: клинообразное тело сечения. Верхнее ребро клина направлено параллельно оси Y и оно расположено в gx, гу, rz (гу игнорируется).

status: управляет протяженностью тела сечения и трактовкой создаваемых многоугольников сечения и новых ребер.

$$\text{status} = j1 + 2*j2 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8$$

j1: для создания граней и ребер многоугольника при рассечении используются реквизиты самого тела.

j2: создаваемые многоугольники сечения трактуются как обычные многоугольники.

j4, j5: определяет ограничения на сечение:

j4 = 0 и j5 = 0: конечное сечение,

j4 = 0 и j5 = 1: полубесконечное сечение,

j4 = 1 и j5 = 1: бесконечное сечение.

j6: в качестве результата выбирается то, что получается при пересечении исходного тела с телом сечения, а не их разность. (Может использоваться только с командой CUTFORM)

j7 : ребра, создаваемые нижней частью тела сечения, будут невидимыми.

j8 : ребра, создаваемые верхней частью тела сечения, будут невидимыми.

gx,гу,rz: определяет направление сечения, если фигура сечения имеет форму призмы, или вершину пирамиды, если тело сечение пирамидальное.

d: определяет расстояние вдоль gx, гу, rz до конца сечения. Если сечение бесконечное, то этот параметр не действует. Если сечение конечное, то тело сечения начинается в начале локальной системы координат и заканчивается на расстоянии d вдоль направления, определенного gx, гу, rz.



Если сечение полубесконечное, то тело сечения начинается в точке, расположенной на расстоянии  $d$  вдоль направления, определенного  $bu$ ,  $gu$ ,  $gz$ , а направление полубесконечного сечения является противоположным направлению, определяемому  $gx$ ,  $gy$ ,  $gz$ .

`mask`: определяет видимость ребер тела сечения.

$mask_i = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 64*j_7$

$j_1$ : многоугольник будет создавать видимое ребро при входе в высекаемое тело.

$j_2$ : продольное ребро фигуры сечения будет видимым.

$j_3$ : многоугольник будет создавать видимое ребро при выходе из высекаемого тела.

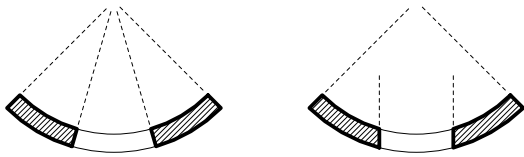
$j_4$ : нижнее ребро фигуры сечения будет видимым.

$j_5$ : верхнее ребро фигуры сечения будет видимым.

$j_7$ : управляет видимостью продольного ребра, которая зависит от точки наблюдения.

## Прямоугольные двери и окна в криволинейных стенах

При размещении дверей/окон в криволинейных стенах боковые стороны проема, образующегося в результате сечения, могут изменяться согласно приведенному ниже рисунку.



Изображенный на рисунке слева оконный проем создается ArchiCAD автоматически при построении отверстия для двери/окна. В этом случае боковые стороны имеют радиальное направление. На рисунке справа изображен проем, создаваемый по команде `WALLHOLE`, выполняющейся в 3D-скрипте двери/окна. Таким образом необходимо, чтобы сам библиотечный элемент был описан с учетом указанных факторов.

Другой момент, который нельзя опустить из внимания при размещении окна/двери в дугообразной стене, это форма самого окна/двери: прямолинейная или дугообразная.



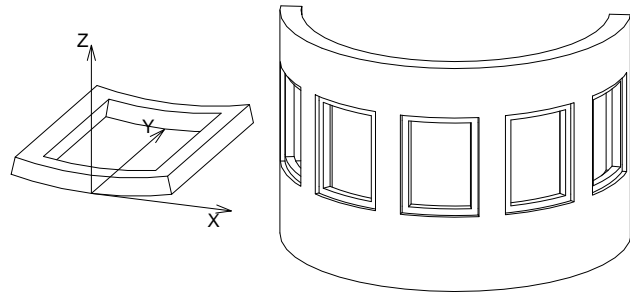
В случае размещения прямолинейного окна/двери, как показано на рисунке выше слева, толщина и ширина объекта и толщина стены тесно взаимосвязаны, так как размеры объекта не должны превышать допускаемые проемом в стене. При использовании дугообразного окна/двери эта проблема отсутствует.

Пример:

```

RESOL 72
ROTX -90
MULY -1
C= 0.12: Z=(360*A)/(2*R_*PI)
Y= (360*C)/(2*R_*PI)
A1= 270+Z/2: A2=270-Z/2
GOSUB 1
ADDZ B
MULZ -1
GOSUB 1
DEL 2
ADDZ C
GOSUB 2
MULX -1
GOSUB 2
END
1:
PRISM 9, C,
  COS(A2)*R_, SIN(A2)*R_+R_, 11,
  COS(A2+Y)*R_, SIN(A2+Y)*R_+R_, 13,
  0, R_, 900,
  0, Z-2*Y, 4009,
  COS(A1)*R_, SIN(A1)*R_+R_, 11,
  COS(A1)*(R_-0.1), SIN(A1)*(R_-0.1)+R_, 11,
  COS(A1-Y)*(R_-0.1), SIN(A1-Y)*(R_-0.1)+R_, 13,
  0, -(Z-2*Y), 4009,
  COS(A2)*(R_-0.1), SIN(A2)*(R_-0.1)+R_, 11
RETURN
2:
PRISM 4, B-2*C,
  COS(A2)*R_, SIN(A2)*R_+R_, 10,
  COS(A2+Y)*R_, SIN(A2+Y)*R_+R_, 15,
  COS(A2+Y)*(R_-0.1), SIN(A2+Y)*(R_-0.1)+R_, 10,
  COS(A2)*(R_-0.1), SIN(A2)*(R_-0.1)+R_, 10
RETURN

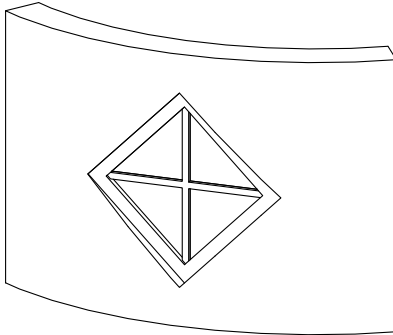
```



## Непрямоугольные двери и окна в криволинейных стенах

Общие положения, изложенные выше для прямоугольных окон и дверей, размещаемых в криволинейных стенах, в данном случае применимы в полной мере.

*Пример:*



```

C=0.1: D=0.025
Z=A/2-SQR(2)*C: Y=A/2-SQR(2)*C-D
ADDY A/2
WALLHOLE 4, 1,
    0, -A/2, 15,
    A/2, 0, 15,
    0, A/2, 15,
    -A/2, 0, 15
PRISM_ 10, 0.1,
    0, -A/2, 15,
    A/2, 0, 15,
    0, A/2, 15,
    -A/2, 0, 15,
    0, -A/2, -1,
    0, -Z, 15,
    Z, 0, 15,
    0, Z, 15,
    -Z, 0, 15,
    0, -Z, -1
ADDZ 0.02
GOSUB 1
ADDZ 0.03
GOSUB 1

```

```
ADDY -Z
SET MATERIAL "Стекло"
ROTZ 45
RECT SQR(2)*Z, SQR(2)*Z
END
1:
PRISM_ 16, 0.03,
      0, -Z, 15,
      D, -Y, 15,
      D, -D, 15,
      Y, -D, 15,
      Z, 0, 15,
      Z, D, 15,
      D, D, 15,
      D, Y, 15,
      0, Z, 15,
      -D, Y, 15,
      -D, D, 15,
      -Y, D, 15,
      -Z, 0, 15,
      -Y, -D, 15,
      -D, -D, 15,
      -D, -Y, 15
RETURN
```

## 2D-варианты

### Вырезание специального проема в стене

По умолчанию размещение окна/двери в стене приводит к вырезанию прямоугольного проема. Размер этого проема в 2D определяется параметром А библиотечного элемента окна/двери. Создание специальных четвертей требует вырезания в стене проемов специальной формы или незначительного расширения стены на плане этажа.

Правильное решение этого вопроса - использование команд WALLHOLE2, WALLBLOCK2, WALLLINE2 и WALLARC2.

### WALLHOLE2

```
WALLHOLE2 n, fill_control, fill_pen, fill_background_pen, fillOrigoX, fillOrigoY, fillAngle,
      x1, y1, s1,
      ...
      xn, yn, sn
```

Определение проема в стене на плане этажа совместно с многоугольником поверхности. Оказывается воздействие только на высекаемую часть стены, видимые многоугольники стены не затрагиваются. Многоугольник поверхности не имеет контура.

Эта команда может использоваться только в 2D-скрипте окна/двери.

Параметризация команды такая же, как и в команде POLY2\_B{2}.

$fill\_control = 2*j2 + 8*j4 + 16*j5 + 32*j6 + 64*j7$

где  $j2, j4, j5$  могут принимать значения 0 или 1.

$j2$  (2): чертить штриховку поверхности на многоугольнике.

$j4$  (8): ориентация локальной штриховки.

$j5$  (16): локальная штриховка должна совпадать с ориентацией стены (начало штриховки находится в начале стены, и ориентация совпадают).

$j6$  (32),  $j7$  (64): определение типа штриховки.

0: штриховка чертежей

32: штриховка сечений;

64: штриховка поверхностей.

Расширение многоугольника стены.

## WALLBLOCK2

```
WALLBLOCK2 n, fill_control, fill_pen, fill_background_pen, fillOrigoX, fillOrigoY, fillAngle,
           x1, y1, s1,
           ...
           xn, yn, sn
```

Определение многоугольника стены (расширение) для плана этажа. Как отсеченные многоугольники стены, так и просматриваемые ее многоугольники будут высекаться определяемым многоугольником. Проемы стены, определенные посредством WALLHOLE2 в другом объекте окна/двери, вырезают многоугольник, созданный командой WALLBLOCK2, а проемы в стене, получаемые из этого же объекта, - нет.

Эта команда может использоваться только в 2D-скрипте объектов окон/дверей.

Параметризация этой команды такая же, как и команды WALLHOLE2.

## WALLLINE2

```
WALLLINE2 x1, y1, x2, y2
```

Определение (расширения) линии стены между двумя точками плана этажа. Проемы, определенные посредством WALLHOLE2 в другом объекте окна/двери, отсекают линию, построенную с помощью команды WALLLINE2, а проемы в стене, получаемые из этого же объекта, - нет.

Эта команда может использоваться только в 2D-скрипте объектов окон/дверей.

Параметризация этой команды такая же, как и команды “LINE2” .

WALLARC2

WALLARC2 *x, y, r, alpha, beta*

Дуга с центром в точке (*x, y*), начинающаяся под углом *alpha* и заканчивающаяся под углом *beta*, с радиусом *r*, которая вычерчивается содержащей ее стеной. Проемы стены, определяемые посредством WALLHOLE2 в другом объекте окна/двери, отсекают дугу, созданную командой WALLARC2, а проемы в стене, получаемые из этого же объекта, - нет.

Эта команда может использоваться только в 2D-скрипте объектов окон/дверей.

Параметризация этой команды такая же, как и команды “ARC2” .

## **СКРИПТ GDL, СОЗДАВАЕМЫЙ ИЗ ПЛАНА ЭТАЖА**

Сохранение плана этажа в виде скрипта GDL или библиотечного элемента приводит к созданию элементов GDL. Вы можете использовать эти скрипты GDL в качестве шаблонов для создания новых библиотечных элементов.

## КЛЮЧЕВЫЕ СЛОВА

### Общие ключевые слова

#### Операторы, функции

FOR, NEXT  
DO, WHILE, ENDWHILE  
REPEAT, UNTIL  
IF, THEN, ELSE, ENDIF  
GOTO  
GOSUB  
RETURN  
END  
EXIT  
PUT  
GET  
USE  
NSP  
CALL, PARAMETERS  
PRINT  
OPEN  
INPUT  
VARTYPE  
OUTPUT  
CLOSE  
DIM  
BREAKPOINT

### Устаревшие ключевые слова

Приведенные ниже ключевые слова являются устаревшим. Они оставлены только для поддержания совместимости с более ранними версиями ArchiCAD.

BAS  
BOX  
FILTER  
GDLBIN  
LIN  
LINE  
NOD  
NODE  
ORIGO  
PARS  
PLOTMAKER  
PLOTTER  
RECT\_  
SFLINE  
TET  
TETRA  
TRI  
WALL\_  
VOCA  
UI\_OK  
UI\_CANCEL

## Ключевые слова только для 3D-фигур

ADDX, ADDY, ADDZ  
ADD  
MULX, MULY, MULZ  
MUL  
ROTX, ROTY, ROTZ  
ROT  
XFORM

HOTSPOT  
LIN\_  
RECT  
POLY, POLY\_  
PLANE, PLANE\_  
CIRCLE  
ARC

BLOCK, BRICK  
CYLIND  
SPHERE  
ELLIPS  
CONE  
PRISM, PRISM\_, CPRISM\_, BPRISM\_, FPRISM\_, SPRISM\_  
SLAB, SLAB\_, CSLAB\_  
CWALL\_, BWALL\_, XWALL\_  
WALLHOLE  
BEAM  
CROOF\_  
ARMC  
ARME  
ELBOW  
EXTRUDE  
PYRAMID  
REVOLVE  
RULED



---

SWEEP  
TUBE, TUBEA  
COONS  
MESH  
MASS  
LIGHT  
PICTURE  
TEXT  
VERT, TEVE  
VECT  
EDGE  
PGON, PIPG  
COOR  
BODY  
BASE  
BINARY

CUTPLANE  
CUTSHAPE  
CUTPOLY  
CUTPOLYA  
CUTEND

DEFINE MATERIAL  
DEFINE TEXTURE  
[SET] MATERIAL  
SHADOW  
MODEL

SECT\_FILL

## Ключевые слова только для 2D-символов

ADD2

MUL2

ROT2

HOTSPOT2

LINE2

RECT2

POLY2, POLY2\_, POLY2\_A, POLY2\_B

ARC2

CIRCLE2

SPLINE2, SPLINE2A

PICTURE2

TEXT2

FRAGMENT2

PROJECT2

DEFINE FILL

DEFINE FILLA

DEFINE LINE\_TYPE

[SET] FILL

[SET] LINE\_TYPE

DRAWINDEX

DRAWING2

DRAWING3

## Ключевые слова для 2D-символов и 3D-фигур

DEL  
[LET]  
RADIUS  
RESOL  
TOLER  
PEN  
DEFINE STYLE  
[SET] STYLE

## Ключевые слова негеометрических скриптов

### Скрипт спецификаций

DATABASE\_SET  
DESCRIPTOR  
COMPONENT  
REF  
SURFACE3D  
VOLUME3D  
  
POSITION  
  
WALLS  
COLUMNS  
BEAMS  
DOORS  
WINDOWS  
  OBJECTS  
PITCHED\_ROOFS  
HIP\_ROOFS  
LIGHTS  
HATCHES  
ROOMS  
MESHERS  
  
DRAWING

BINARYPROP

## **Скрипт параметров**

VALUES

PARAMETERS

LOCK

## **Скрипт интерфейса**

UI\_DIALOG

UI\_PAGE

UI\_BUTTON

UI\_PREV

UI\_NEXT

UI\_GROUPBOX

UI\_SEPARATOR

UI\_PICT

UI\_STYLE

UI\_OUTFIELD

UI\_INFIELD

UI\_FUNCTION

UI\_LINK

UI\_CURRENT\_PAGE

UI\_TOOLTIP



**BITTEST** (x, b)  
**BLOCK** a, b, c  
**BODY** status  
**BPRISM** top\_material, bottom\_material, side\_material,  
n, h, radius, x1, y1, s1, ... xn, yn, sn  
**BREAKPOINT** expression  
**BRICK** a, b, c  
**BWALL** left\_material, right\_material, side\_material,  
height, x1, x2, x3, x4, t, radius,  
mask1, mask2, mask3, mask4,  
n,  
x\_start1, y\_low1, x\_end1, y\_high1, frame\_shown1,  
...  
x\_startn, y\_lown, x\_endn, y\_highn, frame\_shownn,  
m,  
a1, b1, c1, d1,  
...  
am, bm, cm, dm

## C

**CALL** macro\_name\_string [,] **PARAMETERS ALL** [name1=value1, ...  
namen=valuen][[,]**RETURNED\_PARAMETERS** r1, r2, ...]  
**CALL** macro\_name\_string [,] **PARAMETERS** [name1=value1 , ... namen=valuen][[,]  
**RETURNED\_PARAMETERS** r1, r2, ...]  
**CALL** macro\_name\_string [,] **PARAMETERS** [name1=value1 , ... namen=valuen]  
**CEIL** (x) Возвращает наименьшее целое число, которое не меньше x (всегда целое число). (e.g.,  
CEIL(1.23) = 2; CEIL (-1.9) = -1).  
**CIRCLE** r  
**CIRCLE2** x, y, r  
**CLOSE** channel  
**COMPONENT** name, quantity, unit [, proportional\_with, code, keycode, unitcode]  
**CONE** h, r1, r2, alpha1, alpha2

---

**COONS** *n*, *m*, *mask*,  
*x11*, *y11*, *z11*, ... *x1n*, *y1n*, *z1n*,  
*x21*, *y21*, *z21*, ... *x2n*, *y2n*, *z2n*,  
*x31*, *y31*, *z31*, ... *x3m*, *y3m*, *z3m*,  
*x41*, *y41*, *z41*, ... *x4m*, *y4m*, *z4m* 94

**COOR** *wrap*, *vert1*, *vert2*, *vert3*, *vert4*

**COS** (*x*) Returns the cosine of *x*.

**CPRISM** *top\_material*, *bottom\_material*, *side\_material*,  
*n*, *h*, *x1*, *y1*, *s1*, ... *xn*, *yn*, *sn*

**CROOF** *top\_material*, *bottom\_material*, *side\_material*,  
*n*, *xb*, *yb*, *xe*, *ye*, *height*, *angle*, *thickness*,  
*x1*, *y1*, *alpha1*, *s1*,  
...,  
*xn*, *yn*, *alphan*, *sn*

**CSLAB** *top\_material*, *bottom\_material*, *side\_material*,  
*n*, *h*, *x1*, *y1*, *z1*, *s1*, ... *xn*, *yn*, *zn*, *sn*

**CUTFORM** *n*, *method*, *status*,  
*rx*, *ry*, *rz*, *d*,  
*x1*, *y1*, *mask1*,  
...  
*xn*, *yn*, *maskn*

**CUTPLANE** [*x*, *y*, *z* [, *side* [, *status*]]]  
[*statement1* ... *statementn*]  
**CUTEND**

**CUTPLANE{2}** *angle* [, *status*]  
[*statement1* ... *statementn*]  
**CUTEND**

**CUTPOLY** *n*,  
*x1*, *y1*, ... *xn*, *yn*  
[, *x*, *y*, *z*]  
[*statement1*  
*statement2*  
...]

```
statementn]
```

```
CUTEND
```

```
CUTPOLYA n, status, d,  
x1, y1, mask1, ... xn, yn, maskn [,  
x, y, z]  
[statement1  
statement2  
...  
statementn]
```

```
CUTSHAPE d [, status]  
[statement1 statement2 ... statementn]  
CUTEND
```

status: трактует создаваемые многоугольники сечения. Если не указывается (в целях совместимости) принимается значение по умолчанию 3.

```
status = j1 + 2*j2
```

j1: для создания граней и ребер многоугольника при рассечении используются реквизиты самого тела.

j2: создаваемые многоугольники сечения трактуются как обычные многоугольники.

```
CWALL _left_material, right_material, side_material,  
height, x1, x2, x3, x4, t,  
mask1, mask2, mask3, mask4,  
n,  
x_start1, y_low1, x_end1, y_high1, frame_shown1,  
...  
x_startn, y_lown, x_endn, y_highn, frame_shownn,  
m,  
a1, b1, c1, d1,  
...  
am, bm, cm, dm
```

```
CYLIND h, r
```



**D**

```

DATABASE_SET set_name [descriptor_name, component_name, unit_name, key_name, criteria_name,
    list_set_name]
DEFINE EMPTY_FILL name [[,] FILLTYPES_MASK fill_types]
DEFINE FILL name [[,] FILLTYPES_MASK fill_types,] pattern1, pattern2, pattern3, pattern4,
    pattern5, pattern6, pattern7, pattern8,
    spacing, angle, n,
    frequency1, direction1, offset_x1, offset_y1, m1,
    length11, ... length1m,
    ...
    frequencyn, directionn, offset_xn,
    lengthn1, ... lengthnm
DEFINE FILL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE FILL_A parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE FILLA name [,] [FILLTYPES_MASK fill_types,] pattern1, pattern2, pattern3, pattern4,
    pattern5, pattern6, pattern7, pattern8, spacing_x, spacing_y, angle, n, frequency1,
    directional_offset1, direction1,
    offset_x1, offset_y1, m1, length11,
    ...
    length1m, ... frequencyn,
    directional_offsetn, directionn,
    offset_xn, offset_yn, mn,
    lengthn1, ... lengthnm
DEFINE LINE_TYPE name spacing, n,
    length1, ... lengthn
DEFINE LINE_TYPE parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE MATERIAL name [,] BASED_ON orig_name [,] PARAMETERS name1 = expr1 [, ...][[,]
    ADDITIONAL_DATA name1 = expr1 [, ...]]
DEFINE MATERIAL name type, parameter1,parameter2, ... parametern
DEFINE MATERIAL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE SOLID_FILL name [[,] FILLTYPES_MASK fill_types]]
DEFINE STYLE name font_family, size, anchor, face_code

```

---

```
DEFINE STYLE{2} name font_family, size, face_code
DEFINE SYMBOL_FILL name [,][FILLTYPES_MASK fill_types,]
    pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8,
    spacingx1, spacingy1, spacingx2, spacingy2,
    angle, scaling1, scaling2, macro_name [,]
PARAMETERS [name1 = value1, ... namen = valuen]
DEFINE SYMBOL_FILL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE SYMBOL_LINE name dash, gap, macro_name PARAMETERS [name1 = value1, ... namen = valuen]
DEFINE SYMBOL_LINE parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE TEXTURE name expression, x, y, mask, angle
DEL n [, begin_with]
DEL TOP
DESCRIPTOR name [,code, keycode]
DIM var1[dim_1], var2[dim_1][dim_2], var3[ ],
    var4[ ][ ], var5[dim_1][ ],
    var5[ ][dim_2]
DO
    [statement1
    statement2
    ...
    statementn]
DRAWINDEX number
DRAWING2 [expression]
DRAWING3 projection_code, angle, method
DRAWING3{2} projection_code, angle, method [,backgroundColor, origoX, origoY,
    filldirection]
DRAWING3{3} projection_code, angle, method , parts[, backgroundColor, fillOrigoX, fillOrigoY,
    filldirection][[,]
    PARAMETERS name1=value1 , ... namen=valuen]
DRAWING
```

**E**

**EDGE** vert1, vert2, pgon1, pgon2, status

**ELBOW** r1, alpha, r2

**ELLIPS** h, r

**END / EXIT** [v1, v2, ..., vn]

**ENDGROUP**

**EXOR** (or @) Логическое исключаящее ИЛИ приоритет 8

**EXP** (x) Возвращает e в степени x ( $e = 2.7182818$ ).

**EXTRUDE** n, dx, dy, dz, mask, x1, y1, s1,  
..., xn, yn, sn

**F**

**FILE\_DEPENDENCE** "name1" [, "name2", ...]

**FOR** variable\_name = initial\_value TO end\_value [ STEP step\_value ]

**FPRISM** top\_material, bottom\_material, side\_material, hill\_material,  
n, thickness, angle, hill\_height,  
x1, y1, s1,  
...  
xn, yn, sn

**FRA** (x) Возвращает дробную часть x (целое 0, если x - целое число, действительное число в противном случае). (Например,  $FRA(1.23) = 0.23$ ,  $FRA(-1.23) = 0.77$ ).

**FRAGMENT2** fragment\_index,  
use\_current\_attributes\_flag

**FRAGMENT2** ALL, use\_current\_attributes\_flag

## G

**GET** (n)

**GOTO** label

**GOSUB** label

**GROUP** "name"

## H

**HIDEPARAMETER** name1 [, name2, ..., namen]

**HOTARC2** x, y, r, startangle, endangle

**HOTLINE2** x1, y1, x2, y2

**HOTSPOT** x, y, z [, unID [, paramReference, flags] [, displayParam]]

**HOTSPOT2** x, y [, unID [, paramReference, flags][, displayParam]]

**HPRISM** top\_mat, bottom\_mat, side\_mat,  
hill\_mat,  
n, thickness, angle, hill\_height, status,  
x1, y1, s1,  
...,  
xn, yn, sn

## I

**IF** condition **GOSUB** label

**IF** condition **GOTO** label

**IF** condition **THEN** label

**IND** (FILL, name\_string)

**IND** (LINE\_TYPE, name\_string)

**IND** (MATERIAL, name\_string)

**IND** (STYLE, name\_string)

**IND** (TEXTURE, name\_string)

**INPUT** (channel, recordID, fieldID, variable1 [, variable2,...])

**INT** (x) Возвращает целую часть x (всегда целое число). (например,  $\text{INT}(1.23) = 1$ ,  $\text{INT}(-1.23) = -2$ ).

**ISECTGROUP** (g\_expr1, g\_expr2)

**ISECTLINES** (g\_expr1, g\_expr2)

## K

**KILLGROUP** g\_expr

## L

**[LET]** varnam = n

**LGT** (x) Возвращает логарифм x по основанию 10.

**LIGHT** red, green, blue, shadow,  
radius, alpha, beta, angle\_falloff,  
distance1, distance2,  
distance\_falloff [[,] **ADDITIONAL\_DATA** name1 = value1,  
name2 = value2, ...]

**LIN\_** x1, y1, z1, x2, y2, z2

**LINE\_PROPERTY** expr

**LINE2** x1, y1, x2, y2

**LOCK** name1 [, name2, ..., namen]

**LOG** (x) Возвращает натуральный логарифм x.

## M

**MASS** top\_material, bottom\_material, side\_material,  
n, m, mask, h,  
x1, y1, z1, s1,  
...  
xn, yn, zn, sn,  
xn+1, yn+1, zn+1, sn+1,  
...  
xn+m, yn+m, zn+m, sn+m

**MAX** (x1,x2, ... xn) Возвращает наибольшее из задаваемых аргументов.

**MESH** a, b, m, n, mask,  
z11, z12, ... z1m,  
z21, z22, ... z2m,  
...  
zn1, zn2, ... znm

**MIN** (x1,x2, ... xn) Возвращает наименьшее из задаваемых аргументов.

**MODEL** SOLID

**MODEL** SURFACE

**MODEL** WIRE

**MUL** mx, my, mz

**MUL2** x, y

**MULX** mx

**MULY** my

**MULZ** mz

## N

**NEXT** variable\_name

**NOT** (x) Возвращает ЛОЖЬ (=0 - целое), если x - ИСТИНА ( $\neq 0$ ), и ИСТИНА (=1 - целое), если x - ЛОЖЬ (=0) (логическое отрицание).

**NSP**

**NTR** ()

## O

**OPEN** (filter, filename, parameter\_string)

**OR** (or |) Logical inclusive or precedence 7

**OUTPUT** (ch, recordID, fieldID, var1, var2...)

**OUTPUT** channel, recordID, fieldID, expression1 [, expression2, ...]

## P

**PARAGRAPH** name alignment, firstline\_indent,  
left\_indent, right\_indent, line\_spacing [,  
tab\_size1, ...]  
[**PEN** index]  
[[**SET**] **STYLE** style1]  
[[**SET**] **MATERIAL** index]  
'string1'  
'string2'  
...  
'string n'  
[**PEN** index]  
[[**SET**] **STYLE** style2]  
[[**SET**] **MATERIAL** index]  
'string1'  
'string2'  
...  
'string n'  
...  
...

**PARAMETERS** name1 = expression1 [,  
name2 = expression2, ...,  
namen = expressionn]

**PEN** n

**PGON** n, vect, status, edge1, edge2, ... edgen

**PI** Возвращает константу Лудольфа. ( $p = 3.1415926\dots$ ).

**PICTURE** expression, a, b, mask

**PICTURE2** expression, a, b, mask

**PICTURE2{2}** expression, a, b, mask

**PIPG** expression, a, b, mask, n, vect,  
status,  
edge1, edge2, ... edgen

**PLACEGROUP** g\_expr

**PLANE** n, x1, y1, z1, ... xn, yn, zn

---

**PLANE** *n*, *x1*, *y1*, *z1*, *s1*, ... *xn*, *yn*, *zn*, *sn*

**POLY** *n*, *x1*, *y1*, ... *xn*, *yn*

**POLY** *n*, *x1*, *y1*, *s1*, ... *xn*, *yn*, *sn* 61

**POLY2** *n*, *frame\_fill*, *x1*, *y1*, ... *xn*, *yn*

**POLY2** *n*, *frame\_fill*, *x1*, *y1*, *s1*, ... *xn*, *yn*, *sn*

**POLY2\_A** *n*, *frame\_fill*, *fill\_pen*,  
*x1*, *y1*, *s1*, ..., *xn*, *yn*, *sn*

**POLY2\_B** *n*, *frame\_fill*, *fill\_pen*,  
*fill\_background\_pen*,  
*x1*, *y1*, *s1*, ..., *xn*, *yn*, *sn*

**POLY2\_B{2}** *n*, *frame\_fill*, *fill\_pen*,  
*fill\_background\_pen*,  
*fillOrigoX*, *fillOrigoY*,  
*fillAngle*,  
*x1*, *y1*, *s1*, ..., *xn*, *yn*, *sn*

**POLY2\_B{3}** *n*, *frame\_fill*, *fill\_pen*,  
*fill\_background\_pen*,  
*fillOrigoX*, *fillOrigoY*,  
*mxx*, *mxy*, *myx*, *myy*, *x1*, *y1*, *s1*, ..., *xn*, *yn*, *sn*

**POSITION** *position\_keyword*

**PRINT** *expression* [, *expression*, ...]

**PRISM** *n*, *h*, *x1*, *y1*, ... *xn*, *yn*

**PRISM** *n*, *h*, *x1*, *y1*, *s1*, ... *xn*, *yn*, *sn*

**PROJECT2** *projection\_code*, *angle*, *method*

**PROJECT2{2}** *projection\_code*, *angle*, *method* [, *backgroundColor*, *fillOrigoX*,  
*fillOrigoY*, *filldirection*]

**PROJECT2{3}** *projection\_code*, *angle*, *method* [, *parts* [, *backgroundColor*, *fillOrigoX*, *fillOrigoY*,  
*filldirection*][[, ]  
PARAMETERS *name1=value1* , ... *namen=valuen*]

**PUT** *expression* [ , *expression*, ...]

**PYRAMID** *n*, *h*, *mask*, *x1*, *y1*, *s1*, ... *xn*, *yn*, *sn*



**R**

```

RADIUS radius_min, radius_max
RECT a, b
RECT2 x1, y1, x2, y2 114
REF COMPONENT code [, keycode [, numeric_expression]]
REF DESCRIPTOR code [, keycode]
REPEAT
    [statement1
    statement2
    ...
    statementn]
REQ (parameter_string)
REQ (parameter_string)
REQUEST ("ANCESTRY_INFO", expr, name [,
    guid, parent_name1, parent_guid1,
    ...,
    parent_namen, parent_guidn)
REQUEST ("Angular_dimension", "",
    format_string)
REQUEST ("Angular_length_dimension", ""
    format_string)
REQUEST ("Area_dimension", ""
    format_string)
REQUEST ("ASSOCEL_PROPERTIES ", parameter_string, nr_data, data)
REQUEST ("ASSOCLP_PARVALUE", expr, name_or_index, type, flags, dim1, dim2, values)
REQUEST ("Calculation_angle_unit", "",
    format_string)
REQUEST ("Calculation_area_unit", "",
    format_string)
REQUEST ("Calculation_length_unit", "",
    format_string)

```

---

```
REQUEST ("Calculation_volume_unit", "",
        format_string)
REQUEST ("Constr_Fills_display", "", optionVal)
REQUEST ("Elevation_dimension", "",
        format_string)
REQUEST ("FONTNAMES_LIST", "", fontnames)
REQUEST ("Height_of_style", name,
        height [, descent, leading])
REQUEST ("Level_dimension", "",
        format_string)
REQUEST ("matching_properties", type, name1, name2, ...)
REQUEST ("Name_of_program", "", program_name)
REQUEST ("Radial_dimension", "",
        format_string)
REQUEST ("REFERENCE_LEVEL_DATA", "",
        name1, elev1, name2, elev2, name3, elev3)
REQUEST ("Sill_height_dimension", "",
        format_string)
REQUEST ("Style_info", name, fontname [, size, anchor, face_or_slant])
REQUEST ("Window_door_dimension", "",
        format_string)
REQUEST ("WINDOW_DOOR_SHOW_DIM", "", show)
REQUEST ("door_show_dim", "", show)
REQUEST ("window_show_dim", "", show)
REQUEST ("window_door_zone_relev", " ", out_direction)
REQUEST ("Working_angle_unit", "", format_string)
REQUEST ("Working_length_unit", "", format_string)
REQUEST ("Zone_relations", "", category_name, code, name, number [, category_name2,
        code2, name2, number2])
```

---

```

REQUEST ('TEXTBLOCK_INFO',
    textblock_name, width, height)
REQUEST (extension_name, parameter_string, variable1, variable2, ...)
REQUEST (question_name, name | index, variable1 [, variable2,...])
REQUEST (question_name, name | index, variable1 [, variable2,...])
REQUEST{2} ("Material_info", name_or_index,
    extra_param_name,
    value_or_values)
REQUEST{2} ("Material_info", name_or_index,
    param_name, value_or_values)
REQUEST ("HomeDB_info", "", homeDBIntId, homeDBUserId, homeDBName, homeContext)
RESOL n
RETURN
REVOLVE n, alpha, mask, x1, y1, s1, ... xn, yn, sn
RICHTEXT x, y,
    height, 0, textblock_name
RICHTEXT2 x, y, textblock_name
RND (x) Возвращает случайную величину между 0.0 и x ( $x > 0.0$ ) - всегда действительное число.
ROT x, y, z, alpha
ROT2 alpha
ROTX alphax
ROTY alphay
ROTZ alphaz
ROUND_INT (x)
RULED n, mask,
    u1, v1, s1, ... un, vn, sn,
    x1, y1, z1, ... xn, yn, zn
RULED{2} n, mask,
    u1, v1, s1, ... un, vn, sn,
    x1, y1, z1, ... xn, yn, zn

```

**S**

**[SET] FILL** index  
**[SET] FILL** name\_string  
**[SET] LINE\_TYPE** index  
**[SET] LINE\_TYPE** name\_string  
**[SET] MATERIAL** index  
**[SET] MATERIAL** name\_string  
**[SET] STYLE** index  
**[SET] STYLE** name\_string  
**SECT\_ATTRS** fill, fill\_background\_pen,  
fill\_pen, contour\_pen [, line\_type]  
**SECT\_FILL** fill, fill\_background\_pen,  
fill\_pen, contour\_pen  
**SGN** (x) Возвращает целое +1, если x - положительное, возвращает целое -1, если x -  
отрицательное число, в противном случае - 0 (целое число).  
**SHADOW** keyword\_1[, keyword\_2]  
**SIN** (x) Returns the sine of x.  
**SLAB** n, h, x1, y1, z1, ... xn, yn, zn  
**SLAB\_** n, h, x1, y1, z1, s1, ... xn, yn, zn, sn  
**SPHERE** r  
**SPLINE2** n, status, x1, y1,  
angle1, ..., xn, yn, anglen  
**SPLINE2A** n, status, x1, y1, angle1, length\_previous1,  
length\_next1, ...  
xn, yn, anglen, length\_previousn,  
length\_nextn 119  
**SPLIT** (string, format, variable1 [, variable2, ..., variablen])  
**SPRISM** top\_material, bottom\_material, side\_material,  
n, xb, yb, xe, ye, h, angle,  
x1, y1, s1, ... xn, yn, sn

**SPRISM** {2} top\_material, bottom\_material, side\_material,  
 n, xtb, ytb, xte, yte, topz, tangle,  
 xbb, ybb, xbe, ybe, bottomz, bangle,  
 x1, y1, s1, mat1,  
 ...  
 xn, yn, sn, matn

**SQR** (x) Возвращает корень квадратный x (всегда действительное число).

**STR** (numeric\_expression, length, fractions)

**STR** (format\_string, numeric\_expression)

**STR**{2} (format\_string, numeric\_expression [, extra\_accuracy\_string])

**STRLEN** (string\_expression)

**STRSTR** (string\_expression1, string\_expression2)

**STRSUB** (string\_expression, start\_position, characters\_number)

**STW** (string\_expression)

**SUBGROUP** (g\_expr1, g\_expr2)

**SURFACE3D** ( )

**SWEEP** n, m, alpha, scale, mask,  
 u1, v1, s1, ... un, vn, sn,  
 x1, y1, z1, ... xm, ym, zm

**SWEEPGROUP** (g\_expr, x, y, z)

## T

**TAN** (x) Возвращает тангенс x.

**TEVE** x, y, z, u, v

**TEXT** d, 0, expression

**TEXT2** x, y, expression

**TEXTBLOCK** name width, anchor, angle, width\_factor, charspace\_factor, fixed\_height,  
 'string\_expr1' [, 'string\_expr2', ...]

**TOLER** d

**TUBE** n, m, mask,  
u1, w1, s1,  
...  
un, wn, sn,  
x1, y1, z1, angle1,  
...  
xm, ym, zm, anglem

**TUBEA** n, m, mask,  
u1, w1, s1,  
...  
un, wn, sn,  
x1, y1, z1,  
...  
xm, ym, zm

## U

**UI\_BUTTON** type, text, x, y, width, height [, id [, url]]

**UI\_CURRENT\_PAGE** index

**UI\_DIALOG** title [, size\_x, size\_y]

**UI\_GROUPBOX** text, x, y, width, height

**UI\_INFIELD** "name", x, y, width, height [,  
method, picture\_name,  
images\_number,  
rows\_number, cell\_x, cell\_y,  
image\_x, image\_y,  
expression\_imagel, text1,  
...,  
expression\_imagen, textn]

**UI\_INFIELD{2}** name, x, y, width, height [,  
method, picture\_name,  
images\_number,  
rows\_number, cell\_x, cell\_y,  
image\_x, image\_y,  
expression\_imagel, text1,  
...,  
expression\_imagen, textn]

**UI\_INFIELD**{3} name, x, y, width, height [,  
 method, picture\_name,  
 images\_number,  
 rows\_number, cell\_x, cell\_y,  
 image\_x, image\_y,  
 expression\_image1, text1, value\_definition1,  
 ...,  
 expression\_imagen, textn, value\_definitionn]

**UI\_OUTFIELD** expression,x,y,width,height [, flags]

**UI\_PAGE** page\_number

**UI\_PICT** expression, x, y [,width, height][, mask]

**UI\_SEPARATOR** x1, y1, x2, y2

**UI\_STYLE** fontsize, face\_code

**UI\_TOOLTIP**

**UI\_BUTTON** type, text, x, y, width, height [, id [, url]] [ **UI\_TOOLTIP** tooltiptext ]

**UI\_INFIELD** "name", x, y, width, height [,  
 extra parameters ... ] [ **UI\_TOOLTIP** tooltiptext ]

**UI\_INFIELD**{2} name, x, y, width, height [,  
 extra parameters ... ] [ **UI\_TOOLTIP** tooltiptext ]

**UI\_INFIELD**{3} name, x, y, width, height [,  
 extra parameters ... ] [ **UI\_TOOLTIP** tooltiptext ]

**UI\_OUTFIELD** expression, x, y, width, height [, flags] [ **UI\_TOOLTIP** tooltiptext ]

**UI\_PICT** expression, x, y [,width, height [, mask]] [ **UI\_TOOLTIP** tooltiptext ]

**USE** (n)

**V**

**VALUES** "fillparam\_name" [[,] **FILLTYPES\_MASK** fill\_types,] value\_definition1  
 [, value\_definition2, ...]

**VARDIM1** (expr)

**VARDIM2** (expr)

**VARTYPE** (expression)

**VECT** x, y, z

**VERT** x, y, z

**VOLUME3D** ( )

## W

**WALLHOLE** n, status,  
x1, y1, mask1,  
...  
xn, yn, maskn  
[, x, y, z]

**WALLNICHE** n, method, status,  
rx, ry, rz, d,  
x1, y1, mask1,  
...  
xn, yn, maskn

**WHILE** condition DO  
[statement1  
statement2  
...  
statementn]

## X

**XFORM** a11, a12, a13, a14,  
a21, a22, a23, a24,  
a31, a32, a33, a34

**XWALL\_** left\_material, right\_material, vertical\_material, horizontal\_material,  
height, x1, x2, x3, x4,  
y1, y2, y3, y4,  
t, radius,  
log\_height, log\_offset,  
mask1, mask2, mask3, mask4,  
n,  
x\_start1, y\_low1, x\_end1, y\_high1,  
frame\_shown1,  
...  
x\_startn, y\_lown, x\_endn, y\_highn,



```
frame_shownn,  
m,  
a1, b1, c1, d1,  
...  
am, bm, cm, dm,  
status  
XWALL_{2} left_material, right_material, vertical_material, horizontal_material,  
height, x1, x2, x3, x4,  
y1, y2, y3, y4,  
t, radius,  
log_height, log_offset,  
mask1, mask2, mask3, mask4,  
n,  
x_start1, y_low1, x_end1, y_high1,  
sill_depth1, frame_shown1,  
...  
x_startn, y_lown, x_endn, y_highn,  
sill_depthn, frame_shownn,  
m,  
a1, b1, c1, d1,  
...  
am, bm, cm, dm,  
status
```

## Соглашение по именованию параметров

В связи с наличием иерархии подтипов, дочерние библиотечные элементы автоматически наследуют все параметры родительских элементов. (Полную информацию о подтипах и параметрах Вы можете найти в Руководстве пользователя ArchiCAD). Параметры идентифицируются по их именам, поэтому наследуемые параметры и исходные параметры могут иметь одинаковые имена. Поэтому на авторе библиотечного элемента лежит ответственность в том, чтобы избежать конфликтов посредством использования дескриптивных имен параметров с префиксами в виде сокращенных имен библиотечных элементов.

Для параметров обработчика и для определяемых пользователем параметров Graphisoft определил правила именования параметров в своих библиотеках.

**Примечание:** Обработчики придают дополнительную функциональность библиотечным элементам (например, окна и двери высекают проемы в стенах).

Имена параметров с префиксом **ac\_** зарезервированы для специальных параметров, связанных с обработчиками ArchiCAD (например, **ac\_corner\_window**). Познакомьтесь со стандартными шаблонами подтипов библиотеки ArchiCAD на предмет ознакомления с использованием имен.

Стандартные имена параметров Graphisoft начинаются с префикса **gs\_** (например, **gs\_frame\_pen**). Просмотрите библиотечные элементы ArchiCAD для ознакомления с правилами именования их параметров. Используйте эти параметры в Ваших GDL-скриптах, чтобы гарантировать полную совместимость с библиотеками Graphisoft.

Префикс **FM\_** зарезервирован для ArchiFM (например, **FM\_Type**), а **HVAC\_** зарезервирован в ArchiCAD для параметров, относящихся к HVAC - аппаратуре нагревания, вентиляции и кондиционирования воздуха (например, **HVAC\_Manufacturer**).

## РАСШИРЕНИЕ GDL DATA IN/OUT

Расширение “GDL Data In/Out” позволяет получить доступ к простой базе данных с помощью команд GDL. Во всем остальном оно аналогично расширению “Text GDL In/Out”.

### Описание базы данных

База данных - это текстовый файл, в которой записи хранятся в виде отдельных строк. К базе данных можно формулировать запросы и модифицировать ее записи на основе единственного ключа. Ключ и другие элементы разделяются символом (указываемым в команде OPEN).

Длина строк не обязательно должна быть одинаковой, более того, в строках может быть различное количество столбцов.

Если база данных открыта для записи, то должно быть достаточно памяти для размещения дубликата всего файла базы данных.

Открытие и закрытие базы данных может занимать относительно много времени, поэтому не рекомендуем использовать последовательно следующие команды закрытия и открытия базы данных.

Большие базы данных (с более, чем несколько сотен тысяч записей) должны упорядочиваться с помощью ключей.

База данных может быть открыта, опрошена, изменена и закрыта с помощью этого расширения, используя команды OPEN, INPUT, OUTPUT и CLOSE GDL .

## Открытие базы данных

`channel = OPEN (filter, filename, paramstring)`

`filter`: внутреннее имя этого расширения, в данном случае это "DATA".

`filename`: имя открываемой базы данных.

`paramstring`: специфический для расширения параметр, содержащий символы разделителя и режим открытия файла

Открывает базу данных. Если файл базы данных открывается для изменения и такой файл не существует, то создается новый файл. Если файл базы данных открывается для чтения и сам файл не существует, то выводится сообщение об ошибке.

Возвращаемое значение - положительное целое число, которое идентифицирует конкретную базы данных. Это значение в последующем используется в качестве ссылочного номера базы данных.

Если база данных была открыта до выполнения команды открытия, то выполнение этой команды приводит только к созданию номера канала.

**Параметр `paramstring` может содержать следующее:**

**SEPARATOR:** после этого ключевого слова в одинарных кавычках можно указать символ, который Вы хотите использовать в Вашем текстовом файле (как в случае записи, так и чтения) для разделения значений разных столбцов. Специальный вариант - символ табуляции ('\t').

**MODE:** после этого ключевого слова следует режим открытия файла.

Имеется три режима открытия файла:

- RO (только для чтения);
- WA (чтение и изменение);
- WO (чтение и изменение) Если база данных существует, то сделать ее пустой.

**DIALOG:** параметр 'filename' является идентификатором файла, либо он является полным именем пути к нему.

Идентификатор файла - это обычная простая строка, которая будет соответствовать существующему файлу того или иного расширения и ей присваивается значение в процессе стандартного диалога 'Открыть/Сохранить как'. Такое присваивание производится расширением и оно в дальнейшем не будет повторно выдавать сообщения за исключением случая, когда окажется, что файл является недоступным. Если режим открытия является "только для чтения", то расширение выводит диалоговое окно открытия для нахождения существующего документа. В противном случае расширение выводит предупреждающее сообщение для выбора одного из двух вариантов: 'Создать' и 'Найти':

- Найти: поиск существующего файла данных (диалоговое окно открытия);
- Создать: создание нового файла данных (диалоговое окно сохранения).

Всегда проставляете запятые (,) между SEPARATOR, MODE и DIALOG.

**LIBRARY:** Если ключевое слово LIBRARY присутствует в строке параметров, то файл данных должен находиться в загруженной библиотеке.

Если Вы используете отсутствующие ключевые слова, если задан неправильный символ-разделитель или в строке параметров нет ничего, то расширение по умолчанию использует следующую строку:

```
SEPARATOR = '\t', MODE = RO.
```

*Пример:*

```
ch1 = OPEN ("DATA", "file1",  
           "SEPARATOR=';', MODE = RO", DIALOG)  
ch2 = OPEN ("DATA", "file2", "")  
ch3 = OPEN ("DATA", "newfile",  
           "SEPARATOR = '\t', MODE = WA")
```

## Чтение данных из базы данных

```
INPUT (channel, recordID, fieldID, var1  
      [, var2, ...])
```

recordID: значение ключа (числового или строкового типа).

fieldID: номер столбца в заданной записи;  
наименьший номер : 1;  
он ссылается на элемент после ключевого значения).

var1,...: переменные, в которые считываются элементы записи.

Чтение базы данных на основе ключевого значения.

Если находится указанная запись, то происходит считывание элементов, начиная с указанного столбца, и они последовательно размещаются в параметры.

В списке параметров должно быть по крайней мере одно значение. Значения могут быть числового или строкового типа независимо от типов параметров, определенного для них. Возвращаемое значение - количество успешно прочитанных значений.

Если параметров больше, чем значений, то "лишним" параметрам присваивается значение ноль. Если столбец пустой (то есть между символами-разделителями ничего нет), то параметру присваивается значение ноль.

Если требуемая запись отсутствует, то функция возвращает значение (-1).

*Пример:*

```
nr = INPUT (ch1, "key1", 1, v1, v2, v3)  
! ввод трех значений, начиная с первого столбца записи, содержащей ключ "key1"  
PRINT nr, v1, v2, v3
```

## Запись в базу данных

OUTPUT channel, recordID, fieldID, expr1 [, expr2, ...]

recordID: значение ключа (число или строка)

fieldID: flag: укажите 0 (или  $\leq 0$ ) для удаления записи, укажите 1 (или  $> 0$ ) для создания или изменения записи

expr1,...: новое значение элемента найденной или новой записи; в случае удаления эти значения игнорируются.

В случае создания или изменения записи происходит установка записи, принадлежащей заданному ключевому значению. Запись будет содержать задаваемые значения в том порядке, в котором они приводятся в команде. Значения могут быть числового или строкового типа. Должно быть по крайней мере одно выражение expr1 .

В случае удаления запись, содержащая заданное ключевое значение, удаляется из базы данных. Значения выражений expr1 игнорируются, однако по крайней мере одно из них должно присутствовать.

*Пример:*

```
string = "Date: 19.01.1996"
```

```
a = 1.5
```

```
OUTPUT ch2, "keyA", 1, "New record"
```

```
OUTPUT ch2, "keyA", 1, "Modified record"
```

```
OUTPUT ch2, "keyA", 0, 0 ! удаление записи
```

```
OUTPUT ch2, "keyB", 1, a, string
```

## Заккрытие базы данных

CLOSE channel

channel: значение канала.

Закрывает базу данных, идентифицируемую значением канала.

## РАСШИРЕНИЕ GDL DATE TIME

Расширение DateTime позволяет устанавливать различные форматы даты и времени, установленные в Вашем компьютере. Это расширение работает так же, как GDL оперирует файлами. Вы должны открыть канал, прочитать информацию и затем закрыть канал.

Это расширение также доступно посредством команды GDL REQUEST; в этом случае команды OPEN, INPUT и CLOSE вызываются самой командой. Это самый простой способ получить информацию о дате/времени с помощью единственной команды GDL:

```
REQUEST ("DateTime", format, datetimestring)
```

Второй параметр функции REQUEST точно такой же, как и параметр paramstring функции OPEN.

### Открытие канала

```
channel = OPEN (filter, filename, paramstring)
```

filter: внутреннее имя этого расширения; в нашем случае это "DateTime".

filename: не используется (в данном случае нет необходимости открывать какой-либо файл для получения системной даты и времени).

paramstring: специфический для расширения параметр; содержит требуемый выходной формат даты и времени.

Возвращаемое значение - положительное целое число, которое идентифицирует открытый канал. Это значение - номер, ссылающийся на канал. Параметр paramstring может содержать спецификаторы и другие символы.

Спецификаторы следующим образом заменяются значениями даты и времени:

%a	сокращенное название дня недели
%A	полное название дня недели
%b	сокращенное название месяца
%B	полное название месяца
%c	дата и время в формате: 01:35:56 PM Wednesday, March 27, 1996
%d	день месяца в виде десятичного числа (01-31)
%H	часы (24-часовые часы) в виде десятичного числа (00-23)
%I	часы (12-часовые часы) в виде десятичного числа (01-12)
%j	день года в виде десятичного числа (001-366)
%m	месяц в виде десятичного числа (01-12)
%M	минуты в виде десятичного числа (00-59)
%P	указатель AM/PM для 12-часовых часов

%S	секунды в виде десятичного числа (00-61)
%U	номер недели в году (воскресенье - первый день недели) в виде десятичного числа
%w	день недели в виде десятичного числа (0 (воскресенье)-6 (суббота))
%W	номер недели в году (понедельник - первый день недели) в виде десятичного числа (00-53)
%x	дата в формате Wednesday, March 27, 1996
%X	время в формате 01:35:56 PM
%y	год без столетия в виде десятичного числа (00-99)
%Y	год со столетием в виде десятичного числа
%Z	GDL игнорирует этот спецификатор. Согласно стандарту этот спецификатор указывает на вывод временной зоны, если она может быть определена
%%	символ %

*Пример:*

```
dstr = ""
ch = OPEN ("DateTime", "", "%w/%m/%d/%Y, %H:%M%P")
n = INPUT (ch, "", "", dstr)
CLOSE (ch)
PRINT dstr !it prints 3/03/27/1996, 14:36 PM
```

## Чтение информации

```
n = INPUT (channel, "", "", datetimestr)
```

channel: значение канала

datetimestr: значение строкового типа

Производится чтение значения строкового типа, которое представляет дату и/или время в формате, заданном в команде OPEN. Второй и третий параметры не используются (они могут быть пустыми строками или принимать значение 0). Возвращаемое значение - количество успешно прочитанных значений; в данном случае - 1.

## Закрытие канала

```
CLOSE channel
```

Закрывает канал, указываемый параметром channel.

## РАСШИРЕНИЕ GDL FILE MANAGER I/O

Расширение “GDL File Manager In-Out” позволяет организовать в GDL-скрипте просмотр папки на предмет наличия в ней файлов/подпапок.

Указание папки, которую следует просмотреть, производится в команде OPEN.

С помощью команды INPUT производится получение первого/следующего файла/папки в указанной папке.

По команде CLOSE завершается просмотр папки.

### Указание папки

```
channel = OPEN (filter, filename, paramstring)
```

channel : ID папки.

filter : внутреннее имя этого расширения; в нашем случае это "FileMan".

filename : - имя папки просмотра (зависимый от ОС путь) - строка ID папки (в режиме диалога DIALOG - см. далее).

paramstring : специфический для расширения параметр.

Параметры в paramString должны разделяться запятой (,).

1-й параметр: FILES/FOLDERS.

Что Вы хотите найти?

2-й параметр (факультативный): DIALOG.

Указывает, что папка задается посредством ID файла, а не путем файла.

Если это так, то в начальный момент (и в любой момент времени, когда соответствующий путь файла может оказаться неверным) пользователю выводится диалоговое окно для указания строки ID, то есть пути файла, который следует запомнить.

Например, строка

```
folder = OPEN( "FileMan", "c:\\\\
```

приводит к открытию корневой папки на носителе C (на PC) для поиска файла.

### Получение имени файла/папки

```
n = INPUT (channel, recordID, fieldID, var1 [,  
          var2, ...])
```

channel : ID папки (возвращается командой OPEN).

recordID : 0 (зарезервирован для будущего использования).

fieldID : 0 (зарезервирован для будущего использования).

var1, ... : переменная(ые) для получения имени (имен) файла/папки.



n : количество успешно заполненных переменных.

Например, строка

```
n = INPUT (folder, 0, 0, fileName)
```

приводит к нахождению следующего имени файла в указанной папке; возвращает значение 1.

Если больше нет файлов/папок, то переменной n присваивается значение 0.

## Завершение просмотра папки

```
CLOSE (channel)
```

Закрывается папка, указанная значением channel.

*Пример: Просмотр одной папки*

В следующем сегменте кода (как и в разделе 2D-скрипта объекта, например) формируется список файлов в папке, указанной идентификатором MyFavouriteFolder. Сначала пользователь должен присвоить этому идентификатору существующую папку. далее ID MyFavouriteFolder будет представлять эту папку.

```
topFolder = open( "FileMan", "MyFavouriteFolder", "files, dialog" )
```

```
y = 0
```

```
n = input( topFolder, 0, 0, fileName )
```

```
while n = 1 do
```

```
    text2 0, y, fileName
```

```
    y = y - 0.6
```

```
    n = input( topFolder, 0, 0, fileName )
```

```
endwhile
```

```
close( topFolder )
```

## РАСШИРЕНИЕ GDL TEXT I/O

Расширение GDL Text In/Out позволяет открывать внешние текстовые файлы для чтения/записи и для манипулирования ими путем выборки/помещения значений из/в скрипты GDL.

Это расширение интерпретирует строки списка параметров команд OPEN, INPUT, OUTPUT из скрипта GDL.

Это расширение предполагает, что существует папка под именем "ArchiCAD Data Folder", помимо папки ArchiCAD, для определяемых пользователем файлов. (Это имя определено в самом расширении и поэтому не может быть локализовано.) Если папка с таким именем не существует, то расширение создаст ее. Эта папка может содержать подпапки, в которых расширение производит поиск существующих файлов. Расширение может читать и записывать файлы типа TEXT.

### Открытие файла

```
channel = OPEN (filter, filename, paramstring)
```

filter: внутреннее имя этого расширения; в нашем случае это "TEXT".

filename: имя файла, который следует открыть.

paramstring: специфический для расширения параметр, содержащий символы разделителя и режим открытия файла

Открывает файл. Если файл, в который Вы хотите записывать, не существует, то он создается. Если файл открывается для чтения и он отсутствует, то выводится сообщение об ошибке.

Эта функция возвращает целое положительное число, которое идентифицирует открытый файл. Это значение далее используется как ссылочный номер файла.

Параметр paramstring может содержать следующее:

**SEPARATOR:** после этого ключевого слова в одинарных кавычках можно указать символ, который Вы хотите использовать в Вашем текстовом файле (как в случае записи, так и чтения) для разделения значений разных столбцов.

Специальными символами-разделителями являются знак табуляции ('\t') и символ новой строки ('\n').

**MODE:** за этим ключевым словом приводится режим открытия файла. Имеются следующие три режима открытия:

**RO** (только для чтения);

**WA** (только запись, добавление производится в конец файла);

**WO** (только запись, производится перезапись) ранее хранимые в файле данные теряются!

Файл не может открываться одновременно для чтения и записи.

**DIALOG:** если это ключевое слово присутствует, открывается диалоговое окно, в котором указывается имя файла.

**FULLPATH:** если это ключевое слово присутствует, то имя файла интерпретируется как полное имя пути.

**LIBRARY:** если это ключевое слово присутствует, то файл данных должен находиться в загруженной библиотеке.

Между ключевыми словами следует ставить запятую (,).

Если Вы используете отсутствующие ключевые слова, если задан неправильный символ-разделитель или в строке параметров нет ничего, то расширение по умолчанию использует следующую строку: SEPARATOR = '\t', MODE=RO.

*Пример:*

```
ch1 = OPEN ("TEXT", "file1", "SEPARATOR = ';' , MODE = RO")
ch2 = OPEN ("TEXT", "file2", "")
ch3 = OPEN ("TEXT", "file3", "SEPARATOR = '\n' , MODE = WO")
```

## Чтение значений

```
INPUT (channel, recordID, fieldID,
      var1 [, var2, ...])
```

channel: значенике канала.

recordID: номер строки (числовой или строковый).

fieldID: номер столбца в заданной строке.

var1,...: переменные для размещения прочитанных элементов данных.

Считывается столько значений относительно заданной начальной позиции в файле, указанным идентификатором channel, сколько задано параметров var. В списке параметров должен быть, по крайней мере, один параметр var. Функция последовательно размещает считанные значения в параметрах var. Значения могут быть числового или строкового типа не зависимо от типов параметров, определенных для них.

Возвращаемое функцией значение - количество успешно прочитанных значений, при достижении конца файла возвращается (-1).

Номера строк и столбцов должны быть положительными целыми числами; в противном случае выводится сообщение об ошибке.

Если номер строки или столбца неправильный, то команда не выполняется. (n = 0)

Если строка и столбец правильно идентифицируются правильные, то считывается столько значений относительно заданной начальной позиции сколько задано параметров var; если параметров больше, чем значений, то "лишним" параметрам присваивается значение ноль.

Если столбец пустой (то есть, между символами-разделителями ничего нет), то параметру присваивается значение ноль.

*Пример:*

```
nr = INPUT (ch1, 1, 1, v1, v2, v3)    ! ввод трех значений,
                                     ! начиная с первого столбца первой строки
PRINT nr, v1, v2, v3
```

## Запись значений

OUTPUT channel, recordID, fieldID, expr1 [, expr2, ...]

channel: значение канала

recordID: если значение положительное, то за выводимыми значениями следует символ новой строки.

fieldID: это значение не используется.

expr1: выводимое значение.

Записывается столько значений относительно заданной начальной позиции в файле, указанным идентификатором channel, сколько задано выражений expr. Должно быть хотя бы одно выражение. Типы выводимых значений совпадают с типами выражений.

В случае текстового расширения команда OUTPUT либо (в зависимости от режима открытия) заменяет файл, либо добавляет заданные выражения в конец файла, разделяя их символом-разделителем, указанным при открытии файла. В этом случае заданная позиция не интерпретируется.

Параметр recordID используется для того, чтобы в процессе вывода создавать новые строки.

Если recordID положительный, то за выводимыми значениями следует символ новой строки, в противном случае вслед за последним значением выводится символ-разделитель.

*Пример:*

```
string = "Date: 19.01.1996"
```

```
a = 1.5
```

```
OUTPUT ch2, 1, 0, string
```

```
! за string следует новая строка
```

```
OUTPUT ch2, 0, 0, a, a + 1, a + 2
```

```
! после a + 2 следует символ-разделитель
```

```
! без символа новой строки
```

## Заккрытие файла

```
CLOSE channel
```

channel: значение канала

Закрывает базу данных, идентифицируемую значением channel.

*Пример:*

Объект GDL, который копирует содержимое файла "f1" в файлы "f2" и "f3", однако при этом записывает значения "f1", разделенные символом табуляции, в отдельные строки обоих файлов "f2" и "f3".

```
ch1 = open ("ТЕХТ", "f1", "mode = ro")
ch2 = open ("ТЕХТ", "f2", "separator = '\n', mode = wo")
ch3 = open ("ТЕХТ", "f3", "separator = '\n', mode = wo")
i = 1
1:
n = input (ch1, i, 1, var1, var2, var3, var4)
if n <> -1 then
output ch2, 1, 0, var1, var2, var3, var4
output ch3, 1, 0, var1, var2, var3, var4
i = i + 1
goto 1
else
goto 2
endif
2:
close ch1
close ch2
close ch3
end
```

## РАСШИРЕНИЕ PROPERTY GDL

Цель этого расширения - предоставить возможность доступа к базе данных спецификаций ArchiCAD из скриптов GDL. Вы можете открыть таблицы базы данных и формулировать запросы к ее содержимому на подобие того, как это делается в языке SQL. Вы можете получать отдельные записи или множество записей (списки). Отметим, что Вы не можете изменять эту базу данных и добавлять записи к ней.

*Для детального ознакомления с базой данных спецификаций см. документ "Руководство по проведению расчетов", который входит в состав документации ArchiCAD.*

### OPEN

Синтаксис:

```
OPEN ("PROP", "database set name", ["database files"])
```

Параметры:

<database set name>: произвольное имя, которое будет идентифицировать множество файлов базы данных в последующих предложениях OPEN.

<database files>: список текстовых файлов, являющихся частью базы данных спецификаций. Этот параметр факультативный; если уже присвоен <database set name> этим файлам, то можно производить чтение. Порядок файла является фиксированным: <key file> (файл ключей), <component file> (файл компонент), <descriptor file> (файл дескрипторов), <unit file> (файл единиц измерения). Вам нет необходимости указывать полные пути, так как ArchiCAD отыскивает эти файлы в активных библиотеках. Если Вы используете длинные имена файлов, то помещайте их в кавычки (' или ").

Возвращаемое значение: номер канала.

Открывает коммуникационный канал к заданным файлам базы данных. Содержимое файлов базы данных читаются в оперативную память для более быстрого доступа. После открытия базу данных спецификаций нельзя изменять из этого расширения. Хотя обычно в этом нет особой проблемы.

*Примеры:*

```
1) channel = OPEN ("PROP", "sample", "'AC 8_KEY.txt', 'AC 8_COMP.txt', 'AC 8_DESC.txt', 'AC 8_UNIT.txt' ")
```

Открывается база данных, которая состоит из приведенных файлов (то, которые являются файлами базы данных спецификаций ArchiCAD 7.0), и ей присваивается имя "sample". Обратите внимание, что внутри третьего параметра следует использовать различные символы кавычек (можно использовать " и ').

```
2) channel = OPEN ("PROP", "sample", "")
```

Эта команда может использоваться после явного открытия файлов базы данных (как в примере1), но до их закрытия. Это позволяет явно использовать команду в одном месте скрипта Master\_GDL, и позже использовать ее сокращенную форму.

## CLOSE

Синтаксис:

CLOSE (channel\_number)

Возвращаемое значение: отсутствует.

Закрывает ранее открытый коммуникационный канал.

## INPUT

Синтаксис:

INPUT (channel\_number, "query type", "field list", variable1[, ...])

Параметры:

<channel number> - номер коммуникационного канала, возвращенный предыдущей командой OPEN.

<query type> специфицируется запрос, который следует выполнить. Это расширение распознает следующие ключевые слова.

### Запрос на получение одной записи:

**KEY**, <keycode> – отыскивается запись из базы данных ключей, у которой <keycode> является значением реквизита кода ключа. Допустимые поля: KEYCODE, KEYNAME.

**UNIT**, <unitcode> – отыскивается запись из базы данных единиц измерения, у которой <unitcode> является значением реквизита кода единицы измерения. Допустимые поля: UNITCODE, UNITNAME, UNITFORMATSTR.

**COMP**, <keycode>, <code> – отыскивается запись из базы данных единиц измерения, у которой <keycode> является значением реквизита кода ключа, а <code> - значением реквизита кода компоненты. Допустимые поля: KEYCODE, KEYNAME, CODE, NAME, QUANTITY, QUANTITYSTR, UNITCODE, UNITNAME, UNITFORMATSTR

**DESC**, <keycode>, <code> – отыскивается запись из базы данных единиц измерения, у которой <keycode> является значением реквизита кода ключа, а <code> - значением реквизита кода дескриптора. Допустимые поля: KEYCODE, KEYNAME, CODE, NAME, NUMOFLINES, FULLNAME

### Запросы на получения списка записей:

**KEYLIST** – привести все записи базы данных ключей. Допустимые поля: KEYCODE, KEYNAME

**UNITLIST** – привести все записи базы данных единиц измерения. Допустимые поля: UNITCODE, UNITNAME, UNITFORMATSTR

**COMPLIST**[, <keycode>] – список всех записей в базе данных компонент или, если <keycode> задан, то перечисляются только те записи, у которых код ключа равен <keycode>. Допустимые поля: KEYCODE, KEYNAME, CODE, NAME, QUANTITY, QUANTITYSTR, UNITCODE, UNITNAME, UNITFORMATSTR

**DESCLIST**[, <keycode>] – привести все записи базы данных дескрипторов, или, если задан параметр <keycode>, то привести только те записи, чьи коды ключа равны <keycode>. Допустимые поля: KEYCODE, KEYNAME, CODE, NAME, NUMOFLINES, FULLNAME

**COMPDESCLIST**[, <keycode>] – привести все записи базы данных компонент и дескрипторов, или, если задан параметр <keycode>, то привести только те записи, чьи коды ключа равны <keycode>. Допустимые поля: ISCOMP, KEYCODE,

KEYNAME, CODE, NAME, QUANTITY, QUANTITYSTR, UNITCODE, UNITNAME, UNITFORMATSTR, NUMOFLINES, FULLNAME.

Внимательно используйте этот запрос! Если любое из полей является недопустимым в базе данных (например, FULLNAME в базе данных компонент) то оно просто будет опущено из результирующего списка (Вы должны помнить об этом).

<field list> - список реквизитов базы данных, значения которых должны присутствовать в результирующем списке. Если выводится множество записей, то они будут отсортированы в порядке указания полей в этом списке.

Могут использоваться следующие поля:

**KEYCODE** – реквизит кода ключа. Тип: строка. Используется в запросах: KEY, COMP, DESC, KEYLIST, COMPLIST, DESCLIST, COMPDESCLIST.

**KEYNAME** – реквизит имени ключа. Тип: строка. Используется в запросах: KEY, COMP, DESC, KEYLIST, COMPLIST, DESCLIST, COMPDESCLIST.

**UNITCODE** – реквизит кода единицы измерения. Тип: строка. Используется в запросах: UNIT, COMP, UNITLIST, COMPLIST, COMPDESCLIST.

**UNITNAME** – реквизит имени единицы измерения. Тип: строка. Используется в запросах: UNIT, COMP, UNITLIST, COMPLIST, COMPDESCLIST.

**UNITFORMATSTR** – строка формата GDL единицы измерения. Тип: строка. Используется в запросах: UNIT, COMP, UNITLIST, COMPLIST, COMPDESCLIST.

**CODE** – реквизит атрибута компоненты или дескриптора (зависит от запроса). Тип: строка. Используется в запросах: COMP, DESC, COMPLIST, DESCLIST, COMPDESCLIST.

**NAME** – имя компоненты или первая строка записи дескриптора. Тип: строка. Используется в запросах: COMP, DESC, COMPLIST, DESCLIST, COMPDESCLIST.

**QUANTITY** – количество компоненты в виде числа (для проведения расчетов). Тип: число. Используется в запросах: COMP, COMPLIST, COMPDESCLIST.

**QUANTITYSTR** – количество компоненты в виде строки. Тип: строка. Используется в запросах: COMP, COMPLIST, COMPDESCLIST.

**NUMOFLINES** – количество строк в записи дескриптора. Тип: строка. Используется в запросах: DESC, DESCLIST.

**FULLNAME** – вся запись дескриптора. Тип: строка. Используется в запросах: DESC, DESCLIST.

**ISCOMP** – указывает, является ли следующая запись компонентой или дескриптором. Тип: число (1 - компонента, 0 - дескриптор). Используется в запросах: COMPDESCLIST.

<variables> содержится результат выполнения запроса. Вы можете указать несколько переменных, если точно знаете, сколько их будет получено (например, в запросах с единственной результирующей записью), либо Вы можете указать динамический массив. Записи приводятся последовательно.



*Примеры:*

1.  
 INPUT (channel, "KEY, 001", "KEYNAME", keyname)  
 Это простой запрос: имя ключа с кодом '001' помещается в переменную keyname.
2.  
 INPUT (channel, "DESC, 004, 10", "NUMOFLINES, FULLNAME", desc\_txt)  
 Обрабатываются записи дескрипторов с кодами ключей '004' и '10'; в массиве desc\_txt помещаются количество строк текста дескриптора и сам текст дескриптора. Результат следующий:  
 desc\_txt[1] = <количество строк> (число).  
 desc\_txt[2] = <первая строка описания> (строка).  
 ...  
 desc\_txt[<numoflines+1>] = <последняя строка описания>.
3.  
 INPUT (channel, "COMPLIST", "NAME, KEYNAME, QUANTITY", comp\_list)  
 Создается список компонент, отсортированный по полю NAME, затем по KEYNAME и, наконец, по полю QUANTIT. Результат размещается в массиве comp\_list. Результат следующий:  
 complist[1] = <имя1> (строка)  
 complist[2] = <имя ключа 1> (строка)  
 complist[3] = <количество1> (число)  
 complist[4] = <имя2> (строка)  
 ... и.т.д.
4.  
 INPUT (channel, "COMPDESCLIST, 005", "ISCOMP, KEYNAME, NAME, QUANTITY", x\_list)  
 Создается общий список компонент и дескрипторов, для которых keycode> равен '005' Это означает, что приводятся записи из двух таблиц. Результат следующий:  
 x\_list[1] = 0 (число, 0 → это дескриптор).  
 x\_list[2] = <имя1> (строка → дескрипторы не имеют полей <keyname>, поэтому это поле остается пустым).  
 x\_list[3] = 0 (число, дескрипторы не имеют поля количества).  
 ...  
 x\_list[(n\*2)-1] = 1 (число → до сих пор было приведено n-1 дескрипторов, теперь поступают компоненты).  
 x\_list[n\*2] = <имя ключа\_n> (строка).  
 ... и.т.д.

## OUTPUT

Эта команда не реализована в этом расширении, так как база данных спецификаций предназначена только для чтения.

## РАСШИРЕНИЕ GDL XML

Это расширение позволяет чтение, запись и редактирование файлов XML. Оно реализует подмножество интерфейса объектной модели документов (Document Object Model - DOM). XML является текстовым файлом, который использует теги для структуризации данных в виде иерархической системы, наподобие HTML. Документ XML может быть представлен в виде иерархической структуры, вершины которой содержат данные документа. Это расширение распознает следующие типы вершин:

- *Элемент (element)*: то, что располагается между начальным и конечным тегом в документе, или для пустого элемента может существовать тег пустого элемента. Элементы имеют имя, могут содержать атрибуты и обычно, но не обязательно, имеют контент. Вершины типа элемента могут иметь дочерние вершины. Атрибуты располагаются в списке атрибутов, в котором каждый атрибут представлен уникальным именем и значением.
- *Текст (text)*: Последовательность символов. Текст не может иметь дочерних вершин.
- *Комментарий (comment)*: текст, располагающийся между ограничителями комментария: `<!-- сам комментарий -->`. В самом тексте вслед за символом `'-'` должен следовать символ, отличающийся от `'-'`. Это означает, что следующее выражение является допустимым: `<!-- comment --->`. Вершины-комментарии не могут содержать дочерних вершин.
- *Раздел CDATA (CDATASection)*: текст, располагающийся между ограничителями раздела CDATA: `<![CDATA[ сам текст]]>`. В разделе CDATA символы, которые имеют специальное значение в документе, должны представляться так, как они есть. В разделе CDATA единственным распознаваемым специальным символом является конец текста раздела `'>'`. Вершины, являющиеся разделами CDATA, не могут иметь дочерних вершин.
- *Ссылка на сущность (entity-reference)*: ссылка на ранее определенную сущность. Такая вершина может иметь подиерархию, предназначенную только для чтения, и эта подиерархия предоставляет собой значение той сущности, на которую делается ссылка. В процессе синтаксического анализа документа может быть указано, что ссылки на сущности преобразуются в текстовые вершины.

На верхнем уровне имеется точно одна вершина с типом элемента (корень), и несколько вершин типа комментариев. Данное расширение не разрешает наличие вершин типа документа, которые возможны в интерфейсе DOM.

Для каждой вершины этой древовидной структуры имеется имя и строковое значение, смысл которого зависит от типа вершины:

	Имя	Значение
Элемент:	имя тега	"" (пустая строка)
Текст:	"#text"	текстовый контент вершины
Комментарий:	#comment	текстовый контент вершины
Раздел CDAT:	"#cdata-section"	текстовый контент вершины
Ссылка на сущность:	имя сущности, на которую делается ссылка	"" (пустая строка)

Для каждого типа вершины это расширение определяет строковые ключевые слова, которые могут передаваться в расширение в определенных инструкциях:

Элемент:	ELEM
Текст:	TXT
Комментарий:	CMT
Раздел CDATA:	CDATA
Ссылка на сущность:	EREF

Факт успешного или ошибочного завершения выполнения команд OPEN, INPUT или OUTPUT может быть определен по инструкции GetLastError команды INPUT.

## Открытие документа XML

Команда OPEN:

channel = **OPEN** (filter, filename, parameter\_string)

filter: расширение файла. В нашем случае - 'XML'.

filename: имя и путь файла, который следует открыть (или создать) или имя идентификатора, если файл открыт с помощью диалогового окна и расположение файла указывается пользователем.

parameter\_string: последовательность символьных признаков, которые определяют режим открытия файла:

- **'r'**: открытие в режиме чтения. В общем случае может использоваться только команда INPUT.
  - **'e'**: ссылки на сущности не транслируются в текстовые вершины в дереве. Без этого признака в структуре документа отсутствуют ссылки на сущности.
  - **'v'**: при чтении и записи производится проверка на достоверность. Если в документе имеется DTD, то структура документа должна соответствовать DTD. Без этого признака правильно структурированный но неверный документ может быть прочитан и записан без каких-либо сообщений об ошибках.
  - **'n'**: создание нового файла. Если файл существует, то процедура открытия будет безуспешной. (После OPEN первой выполненной инструкцией должна быть CreateDocument.)
  - **'w'**: заменить файл пустым документом, если он (файл) существует. Если он отсутствует, то создается новый файл. (После OPEN первой выполненной инструкцией должна быть CreateDocument.)
  - **'d'**: файл получается из диалогового окна пользователя. При последующей работе этот файл будет ассоциироваться идентификатором, заданным в параметре filename команды OPEN. (Если этот идентификатор уже ассоциирован с файлом, то диалоговое окно не открывается для пользователя.)
  - **'f'**: параметр filename содержит полный путь.
  - **'l'**: файл находится в загруженной библиотеке.
- channel: используется для идентификации данного подключения в последующих командах ввода/вывода.

Если Вы хотите открыть существующий файл XML для его изменения, то в строке параметров не следует использовать признаками 'r', 'n' и 'w'. Должен использоваться только один из признаков 'd', 'r' и 'l'. Если ни один из этих признаков не используется, то считается, что filename - это путь относительно папки документов пользователя.

## Чтение документа XML

DOM - это объектно-ориентированная модель, которая не может быть непосредственно приспособлена к языкам, подобным BASIC, каким является GDL. Для представления вершин в иерархическом дереве мы определяем дескрипторы расположения. Если мы хотим пройти по вершинам дерева, то сначала следует запросить у расширения новый дескриптор расположения. Сначала дескриптор расположения указывает на корневой элемент. Дескриптор - это по сути 32-битный идентификационный номер, значение которого не представляет интереса для скрипта GDL. Та позиция, на которую этот дескриптор указывает, может изменяться по мере перемещения от одной вершины дерева к другой.

Команда INPUT:

```
INPUT (ch, recordID, fieldID, var1, var2...)
```

ch: канал, возвращаемый командой OPEN.

recordID: имя инструкции плюс параметры.

fieldID: обычно положительный дескриптор; var1, var2,...: факультативный список переменных, в которых размещаются отыскиваемые данные.

Инструкции команды INPUT:

**1 GetLastError:** отыскивает результат последней операции.

recordID: "GetLastError".

fieldID: игнорируется.

Возвращаемые значения:

var1: код ошибки/ок

var2: текст, объясняющий ошибку/ок.

**2 NewPositionDesc:** запрос нового дескриптора расположения.

recordID: "NewPositionDesc"

fieldID: игнорируется

Возвращаемое значение: var1: новый дескриптор расположения (первоначально ссылается на корень).

**3 CopyPositionDesc:** запрос нового дескриптора расположения, начальная вершина которого берется из другого дескриптора.

recordID: "CopyPositionDesc"

fieldID: существующий дескриптор расположения.

Возвращаемое значение: var1: новый дескриптор расположения (первоначально ссылается на то, на что ссылается дескриптор, заданный в fieldID).

**4 ReturnPositionDesc:** когда дескриптор расположения больше не нужен.

recordID: "ReturnPositionDesc"

fieldID: дескриптор расположения..

Используйте эту инструкцию в том случае, когда больше не нужен дескриптор расположения, полученный из инструкций NewPositionDesc или CopyPositionDesc.

##### 5 **MoveToNode**: изменение дескриптора расположения. (и поиск данных новой вершины)

Эта инструкция может использоваться для навигации по дереву иерархии.

recordID: "MoveToNode searchmode nodename nodetype nodenumber"

fieldID: дескриптор расположения.

searchmode (или movemode): параметр nodename должен содержать путь, который определяет вершину-элемент или ссылку на сущность в документе xml.

Путь задается относительно вершины, заданной в fieldID. Ограничителем является символ ':' (который в противном случае является допустимым символом в имени элемента, поэтому этот вариант действителен не во всех случаях). Строка '..' в пути означает возврат к родительской вершине. Начальная вершина может отличаться от вершины-элемента или ссылки на сущность; в этом случае путь должен начинаться с '..' для возврата на шаг назад. Если на одном и том же уровне имеется несколько вершин с одним и тем же именем, то выбирается первая из них.

Для следующих вариантов перемещения остальные параметры не должны присутствовать:

ToParent: перемещение к родительской вершине относительно вершины, заданной в fieldID.

ToNextSibling: перемещение к следующей вершине того же уровня.

ToPrevSibling: перемещение к предыдущей вершине того же уровня.

ToFirstChild: перемещение к дочерней вершине относительно вершины, заданной в fieldID.

ToLastChild: перемещение к последней дочерней вершине относительно вершины, заданной в fieldID.

Для следующих вариантов поиска остальные параметры могут присутствовать, однако в случае их отсутствия для них имеются значения по умолчанию:

FromNextSibling: поиск начинается со следующей вершины того же уровня и производится в прямом направлении.

FromPrevSibling: поиск начинается с вершины, расположенной перед fieldID производится в обратном направлении на том же уровне.

FromFirstChild: поиск начинается с первой дочерней вершины относительно вершины fieldID и производится в прямом направлении.

FromLastChild: поиск начинается с последней дочерней вершины относительно вершины fieldID и производится в обратном направлении.

nodename: поиск принимает во внимание только те вершины, чьи имена или значения совпадают с nodename. Символы \* и ? , имеющиеся в nodename, рассматриваются как групповые (подстановочные) символы . Для элементов и ссылочных сущностей производится сравнение имен, а для текстов, комментариев и разделов CDATA сравниваются значения. Значение по умолчанию: \*

nodetype: поиск принимает во внимание только вершины тех типов, которые допускаются nodetype. Символ \* обозначает, что допустимы все типы. В противном случае ключевые слова, указывающие типы, объединяются символом "+" для образования типа вершины (это должно быть одно слово без пробелов, например TXT+CDATA.) Значение по умолчанию: \*

nodenumber: Если имеется несколько совпадающих вершин, то здесь указывается последовательный номер той вершины, которая выбирается в результате поиска. (Начальное значение 1) Значение по умолчанию: 1

Возвращаемые значения:

var1: имя вершины,

var2: значение вершины,

var3: ключевое слово типа вершины.

*Пример:*

Мы хотим переместиться назад на том же уровне ко 2-й вершине, которая является элементом или ссылкой на сущность, и имя которой начинается на K:

```
INPUT (ch, "MoveToNode FromPrevSibling K* ELEM+EREF 2", posDesc, name, val, type)
```

**6 GetNodeData:** поиск данных заданной вершины.

recordID: "GetNodeData"

fieldID: дескриптор расположения.

Возвращаемые значения:

var1: имя вершины,

var2: значение вершины,

var3: ключевое слово типа вершины

**7 NumberOfChildNodes:** дает количество дочерних вершин заданной вершины.

recordID: "NumberOfChildNodes nodetype nodename"

fieldID: дескриптор расположения

Следующие факультативные параметры могут сузить множество принимаемых во внимание дочерних вершин:

nodetype: допустимые типы вершин; определяется так же, как и в инструкции MoveToNode

nodename: допустимые имена или значения вершин; names определяется так же, как и в инструкции MoveToNode.

Возвращаемые значения:

var1: количество дочерних вершин.

**8 NumberOfAttributes:** отыскивает количество атрибутов вершины-элемента.

recordID: "NumberOfAttributes attrname"

fieldID: дескриптор расположения (должен ссылаться на вершину-элемент).

attrname: если присутствует, то сужает множество принимаемых в расчет атрибутов, так как будут подсчитываться только те вершины, имена которых (а не значения) совпадают с attrname. В attrname символы \* и ? рассматриваются как групповые символы.

Возвращаемое значение:

var1: количество атрибутов.

**9 GetAttribute:** возвращает данные атрибута вершины-элемента.

recordID: "GetAttribute attrname attrnumber"

fieldID: дескриптор расположения (должен ссылаться на вершину-элемент).

Факультативные параметры:

attrname: задается имя атрибута. Символы \* и ? рассматриваются как групповые символы. Значение по умолчанию: \*

attrnumber: если имеется несколько атрибутов, совпадающих по attrname, то attrnumber указывает, какой по порядку атрибут должен быть принят в расчет. (Нумерация начинается с 1.) Значение по умолчанию: 1

Возвращаемые значения:

var1: значение атрибута,

var2: имя атрибута.

**10 Validate:** проверяет правильность документа.

Проверка правильность документа не производится при выполнении инструкций изменения документа. Она производится при их записи назад в файл на диске, если был использован признак 'v' в строке режима открытия документа. Проверка правильности может быть выполнена в любой момент времени с помощью инструкции Validate. Однако, следует помнить, то выполнение этой инструкции требует довольно много времени и памяти, поэтому не рекомендуем проводить такие проверки после каждого изменения документа.

recordID: "Validate".

fieldID: игнорируется.

## Изменение документа XML

**OUTPUT** (ch, recordID, fieldID, var1, var2...)

ch: канал, возвращаемый командой OPEN.

recordID: имя инструкции плюс параметры.

fieldID: обычно дескриптор расположения.

var1, var2,...: дополнительные входные данные.

**Инструкции команды OUTPUT.** Большинство из инструкций команды OUTPUT являются недопустимыми для файлов, открытых в режиме чтения.

**1 CreateDocument:**

recordID: "CreateDocument"

fieldID: игнорируется.

var1: название документа. Это также будет именем тега корневого элемента.

CreateDocument допускается только в том случае, когда файл открыт в режиме нового файла или режиме замены существующего файла. В этих режимах данная инструкция должна быть первой выполненной инструкцией, так как именно по ней создается документ XML.

**2 NewElement:** вставляет в документ новую вершину элемента.

recordID: "NewElement insertpos"

fieldID: дескриптор расположения, относительно которого вставляется новая вершина.

var1: имя нового элемента (имя тега элемента).

insertpos может принимать следующие значения:

AsNextSibling: новый элемент вставляется после той позиции, которая указана в fieldID.

AsPrevSibling: новый элемент вставляется перед той позиции, которая указана fieldID

AsFirstChild: новый элемент вставляется в качестве первой дочерней вершины по отношению к той, которая указана в fieldID (она должна быть вершиной элемента).

AsLastChild: новый элемент вставляется в качестве последней дочерней вершины по отношению к той, которая указана в fieldID (она должна быть вершиной элемента).

**3 NewText:** в документ вставляется новая текстовая вершина.

recordID: "NewText insertpos"

fieldID: дескриптор расположения.

var1: вставляемый текст.

*См. также инструкцию NewElement.*

**4 NewComment:** в документ вставляется новая вершина комментария.

recordID: "NewComment insertpos"

fieldID: дескриптор расположения.

var1: вставляемый текст комментария.

*См. также инструкцию NewElement.*

**5 NewCDATASection:** в документ вставляется новая вершина раздела CDATA.

recordID: "NewCDATASection insertpos"

fieldID: дескриптор расположения.

var1: вставляемый текст раздела CDATA.

*См. также инструкцию NewElement.*

**6 Copy:** создает копию поддерева документа, расположенного под указанной вершиной.

recordID: "Copy insertpos"

fieldID: дескриптор расположения, относительно которого вставляется поддерево.

var1: дескриптор расположения, дающий вершину того поддерева, которое копируется.

insertpos: как и в инструкции NewElement.

копируемое поддерево остается прежним. Дескрипторы расположений, указывающие на вершины копируемого поддерева, будут указывать на те же вершины после копирования.



**7 Move:** перемещает некоторое поддерево документа в некоторое другое место.

recordID: "Move insertpos"

fieldID: дескриптор расположения, относительно которого вставляется поддерево.

var1: дескриптор расположения, дающий вершину того поддерева, которое перемещается.

insertpos: как и в инструкции NewElement.

Исходное поддерево удаляется. Дескрипторы расположений, указывающие на вершины перемещаемого поддерева, будут указывать на те же вершины нового расположения поддерева.

**8 Delete:** удаляет из документа вершину и его поддерево.

recordID: "Delete"

fieldID: дескриптор расположения, определяющий удаляемую вершину.

Дескрипторы расположений, указывающие на вершины удаленного поддерева, становятся недействительными.

**9 SetNodeValue:** изменяет значение вершины.

recordID: "SetNodeValue"

fieldID: дескриптор расположения; он должен ссылаться на вершину типа текста, комментарий или раздела CDATA.

var1: новое текстовое значение вершины.

**10 SetAttribute:** изменяет атрибут вершины-элемента или создает новый.

recordID: "SetAttribute"

fieldID: дескриптор расположения; он должен ссылаться на вершину-элемент.

var1: имя атрибута.

var2: текстовое значение атрибута.

Если элемент уже содержит атрибут с указанным именем, то заменяется его значение, в противном случае в список атрибутов элемента добавляется новый.

**11 RemoveAttribute:** удаляет атрибут вершины-элемента.

recordID: "RemoveAttribute"

fieldID: дескриптор расположения; он должен ссылаться на вершину-элемент.

var1: имя удаляемого атрибута.

**12 Flush:** записывает текущий документ назад в файл.

recordID: "Flush"

fieldID: игнорируется.

Если файл был открыт в режиме проверки его правильности, то тогда сохраняется только правильный документ.

**13 ChangeFileName:** ассоциирует с текущим документом другой файл.

recordID: "ChangeFileName"

fieldID: новый путь файла.

var1: указывает, как интерпретировать fieldID. Если var1 содержит пустую строку, то fieldID содержит путь относительно папки документов пользователя. 'd' указывает, что место расположения файла получается от пользователя посредством диалогового окна (см. признаки режима открытия файла в *“OPEN” на стр. 310*). 'l' указывает, что файл берется из загруженных библиотек. 'f' указывает, что fieldID содержит полный путь.

К этой инструкции можно обратиться, даже если файл был открыт в режиме только чтения. В этом случае после выполнения этой инструкции документ теряет атрибут "только чтение", поэтому он может быть изменен и сохранен в новом месте расположения файла.

Коды ошибок и соответствующие им сообщения:

- 0: "Ок"
- 1: "Отказ в инициализации расширения"
- 2: "Недостаточно памяти"
- 3: "Ошибочный строковый параметр"
- 4: "Ошибка диалога файла"
- 5: "Файл не существует"
- 6: "Ошибка синтаксического анализа XML"
- 7: "Ошибка операции файла"
- 8: "Файл уже существует"
- 9: "Этот канал не открыт"
- 10: "Синтаксическая ошибка"
- 11: "Ошибка открытия"
- 12: "Неправильный дескриптор расположения"
- 13: "Неправильный тип вершины для этой операции"
- 14: "Такая вершина не найдена"
- 15: "Внутренняя ошибка"
- 16: "Ошибка параметра"
- 17: "Такой атрибут не найден"
- 18: "Неправильный документ XML"
- 19: "Неуправляемая исключительная ситуация"
- 20: "Документ только для чтения"
- 21: "Не разрешается создавать документ"
- 22: "Отказ в создании документа"
- 23: "Отказ в установке значения вершине"
- 24: "Перемещение не допускается"
- 25: "Удаление не допускается"
- 26: "Не допускается SetAttribute"
- 27: "Ошибка файла формата"
- 28: "Вставка (или копирование) не допускается"
- 29: "Отказ в создании вершины"
- 30: "Плохая строка"
- 31: "Недопустимое имя"

# ПРЕДМЕТНЫЙ УКАЗАТЕЛЬ

## Цифры

- 2D-символ
  - создание ~ окна/двери 257
- 2D-скрипт 14
- 3D-сетка
  - равноотстоящая ~ 58
- 3D-скрипт 14

## А

- ABS 199
- ACS 200
- ADD 29
- ADD2 27
- ADDGROUP 110, 114
- ADDITIONAL\_DATA 179
- ADDX 28
- ADDY 28
- ADDZ 28
- AND 198
- ARC2 123
- ArchiCAD 14
  - смета компонент в ~ 184
  - смета элементов в ~ 133
- ArchiFM 14
- ARMC 59
- ARME 60
- ASN 200
- ATN 200

## В

- BackgroundColor 130
- BEAM 55
- BINARY 14, 116

- BINARYPROP 14, 183
- BITSET 201
- BITTEST 201
- BLOCK 33
- BODY 100
- BPRISM\_ 40
- BREAKPOINT 214
- BRICK 33
- BWALL\_ 49

## С

- CALL 218
- CEIL 199
- CIRCLE2 124
- CLOSE 222
- COMPONENT 182, 183
- CONE 35
- COONS 86
- COOR 98
- COS 200
- CPRISM\_ 39
- CSLAB\_ 47
- CUSTOM 185
- CUTFORM 108
- CUTPLANE 102
- CUTPOLY 104
- CUTPOLYA 106
- CUTSHAPE 108
- CWALL\_ 47
- CYLIND 33

## Д

- DATABASE\_SET 181

- DEFINE FILL 180, 281
- DEFINE FILL\_A 180
- DEFINE FILLA 171
- DEFINE LINE\_TYPE 175, 180
- DEFINE MATERIAL 163, 180
- DEFINE STYLE 176
- DEFINE SYMBOL\_FILL 173, 180, 282
- DEFINE SYMBOL\_LINE 175, 180
- DEL 31
- DEL TOP 32
- DESCRIPTOR 182
- DIALOG 306
- DIM 195
- DO 210
- DRAWINDEX 161
- DRAWING 184
- DRAWING2 133
- DRAWING3 133
- DRAWING3{2} 134

## Е

- ELBOW 60
- ELLIPS 34
- ELSE 212
- END 23, 214
- ENDGROUP 114
- ENDIF 212
- ENDWHILE 210
- EXIT 23
- EXOR 198
- EXP 200
- EXTRUDE 66

- F**  
 FILE\_DEPENDENCE 180, 221, 283  
 FILL 161, 168, 219  
 FILLA 171  
 FOR 209  
 FPRISM\_ 41  
 FRA 199  
 FRAGMENT2 14, 128  
 FULLPATH 306
- G**  
 GET 215  
 GOSUB 212, 213  
 GOTO 212, 213  
 GROUP 114
- H**  
 HIDEPARAMETER 187  
 HOTARC2 140  
 HOTLINE2 140  
 HOTSPOT 135  
 HOTSPOT2 119, 135  
 HPRISM\_ 43
- I**  
 IF 212  
 IND 202, 251  
 INPUT 221  
 INT 199  
 ISECTGROUP 110, 114  
 ISECTLINES 111, 114
- K**  
 KILLGROUP 115
- L**  
 LET 153  
 LGT 200  
 LIBRARY 306  
 LIGHT 91  
 LINE\_PROPERTY 121, 157, 285  
 LINE\_TYPE 162, 175, 219  
 LINE2 119  
 LOCK 187  
 LOG 201
- M**  
 MASS 89  
 MASTER\_GDL 21, 25, 162, 185  
 MASTEREND\_GDL 21  
 MATERIAL 163, 219  
 MATERIAL. 158, 163  
 MAX 201  
 MESH 58  
 MIN 201  
 MOD 198  
 MODE 306  
 MODEL 157, 219  
 MODEL SURFACE 113  
 MUL 29  
 MUL2 28  
 MULX 29  
 MULY 29  
 MULZ 29
- N**  
 NEXT 209  
 NOT 201  
 NSP 215  
 NTR 32
- O**  
 OPEN 221  
 OUTPUT 221
- P**  
 PARAMETERS 186  
 PGON 97  
 PI 200  
 PICTURE 14  
 PICTURE2 14, 127  
 PICTURE2{2} 127  
 PIPG 98  
 PLACEGROUP 115  
 POLY2 120  
 POLY2\_ 121  
 POLY2\_A 122  
 POLY2\_B 122  
 POSITION 184  
 PRINT 220  
 PRISM 35  
 PRISM\_ 36  
 PROJECT2 129  
 PROJECT2{2} 129  
 PYRAMID 69
- R**  
 RADIUS 154, 219  
 RANGE 185  
 RECT2 120  
 REF 182  
 REQ 202, 245  
 REQUEST 202, 246  
 RESOL 219  
 RETURN 213  
 REVOLVE 71  
 RICHTEXT 95, 291

RICHTEXT2 128, 291  
 RND 201  
 RO 306  
 ROT 30  
 ROT2 28  
 ROTX 30  
 ROTY 30  
 ROTZ 30  
 RULED 74  
 RULED{2} 75

**S**

SECT\_FILL 159  
 SEPARATOR 306  
 SET FILL 161  
 SET LINE\_TYPE 162  
 SET MATERIAL 158  
 SET STYLE 157  
 SGN 199  
 SHADOW 160, 219  
 SIN 200  
 SOLID 157  
 SPHERE 34  
 SPLINE2 124  
 SPLINE2\_A 126  
 SPLIT 205  
 SPRISM\_ 43  
 SQR 199  
 STEP 185, 209  
 STR 202  
 STR{2} 202  
 STRLEN 206  
 STRSTR 206  
 STRSUB 207  
 STW 206  
 STYLE 157, 176, 219

SUBGROUP 110, 114  
 SURFACE 157  
 SURFACE3D 183  
 SWEEP 77  
 SWEEPGROUP 111, 116

**T**

TAN 200  
 TEVE 96  
 TEXT 94  
 TEXT2 127  
 TEXTURE 166  
 THEN 212  
 TO 209  
 TOLER 156, 219  
 TUBE 80  
 TUBEA 84

**U**

UI\_BUTTON 188  
 UI\_DIALOG 187  
 UI\_GROUPBOX 189  
 UI\_INFIELD 191  
 UI\_OUTFIELD 190  
 UI\_PAGE 187  
 UI\_SEPARATOR 189  
 UI\_STYLE 190  
 UNTIL 211  
 USE 215

**V**

VALUES 185  
 VARDIM1 196  
 VARDIM2 196  
 VARTYPE 221  
 VECT 96

VERT 96

**W**

WA 306  
 WALLHOLE 260  
 WALLNICHE 264  
 WHILE 210  
 WIRE 157  
 WO 306

**X**

XFORM 30  
 XWALL\_{2} 53

**Б**

База данных 181  
 База-поверхность 112  
 Балка 55  
 Библиотечный элемент 13  
 Бинарные 2D-данные 14  
 Бинарные 3D-данные 14  
 Бинарные данные спецификаций 14

**В**

Вектор 95  
     определение ~ 96  
 Векторная штриховка 170  
 Вершина 95  
 Восклицательный знак в GDL-скрипте 23  
 Вывод аргументов 220

**Г**

Геометрические примитивы 20  
 Гибридное тело 112  
 Глобальная переменная 21, 25  
 Глобальное начало координат 20

**Д**

Двоеточие в GDL-скрипте 23

Дескриптор 14

определение ~ 182

ссылка на ~ 182

Дуга 123

**З**

Заккрытие файла 222

Запоминание значений в буфере значений  
215

Запятая в GDL-скрипте 23

Значения маски 142

**И**

Идентификатор 24

Изменение типа элемента в смете 184

ИЛИ 198

Импорт значений из файла 221

Источник света 91

**К**

Кавычки в GDL-скрипте 24

Каркасная база 112

Каркасная модель 157

Каркасное тело 112

Квадратные скобки в GDL-скрипте 26

Команды средней сложности 16

Комментарий 14

Компонента

определение ~ 182

ссылка на ~ 183

Конец определения скрипта 214

Конец стены 227

Константа Лудольфа 200

Криволинейная призма 40

Криволинейная стена 49

Крыша

наклонная ~ 55

**М**

Макровывоз 19

определение ~ 218

Макрообъект 218

Макрос

~ для окна/двери 258

Массив параметров 26

Масштабирование локальной системы  
координат 29

Метка 23

Многоугольник 95, 97, 120

обобщенный ~ 122

~ рисунка 98

**Н**

Набор символов GDL 23

Натуральный логарифм 201

Непрямоугольная дверь и окно

~ в криволинейной стене 267

~ в прямолинейной стене 260

**О**

Образец штриховки

простой ~ 169

Обращение к запросу 202

Объем 3D-фигуры 183

Объемная база 112

Объемное тело 112

Окружность 124

Операции с файлами 220

Определение покрытия 163

Определение текстуры 166

Основная система координат 20

Основной скрипт 13

Основные синтаксические элементы 23

Открытие файла 221

Отрезок (линия) 119

**П**

Параллелепипед 33

Параметр 14, 25

блокировка значения ~ 187

буфер ~ 214

изменение значения ~ в GDL 186

производный тип 26

простой тип 25

~ в скрипте GDL 19

Параметры бревенчатой стены 51

Переменная 24, 26

Перемещение локальной системы координат  
28

Перо 219

Пирамида 69

Плоская ломаная линия

окружность по ее центру и радиусу 149

отрезок по относительным координатам  
второй точки 144

Объем 3D-фигуры 183

Площадь поверхности 3D-фигуры 183

Поверхность вращения 71

Поверхность, создаваемая из ломаных

~ в плоскости и пространстве 75

~ перемещением образующей ломаной  
по направляющей ломаной в  
пространстве 77, 80, 84

Поворот локальной системы координат 30

Подпрограмма 212, 213

Подсказка 26

Полная матрица преобразований 30

Полуэллипсоид 34

- Пользовательская глобальная переменная 25
- Порядок построения элементов 161
- Правила маскирования
- ~ для 3D-сеток 58
  - ~ для призм 141
- Предложение 23
- Предложения условного и безусловного перехода 18
- Преобразование координат 27
- ~ средней сложности 17
  - основы ~ 16
- Призма 35
- ~ общего вида 66
  - ~ с непараллельной верхней гранью 43
  - ~ с усечением 41
  - косая ~ 46
  - обобщение ~ 39
  - обобщенная косая ~ 46
- Присвоение значения 153
- Проекция 3D-скрипта в 2D-символ 129
- Простейшие команды 15
- Простые фигуры 15
- Профессиональное использование GDL 19
- Прямоугольник 120
- Прямоугольное окно или дверь
- ~ в прямолинейной стене 258
- Р**
- Растровый рисунок 170
- Расширение текста 298
- Ребро 95, 97
- Реквизит
- определение ~ 21
- Рисунок 93, 127
- Рисунок образца 14
- С**
- Сбрасывание счетчика
- ~ для примитивных элементов 102
- Сглаживание цилиндрических элементов
- ~ посредством разрешающей способности 155
  - ~ путем аппроксимации 156
  - ~ радиусом 154
- Сечение многоугольником 104, 106
- Система координат
- локальная ~ для примитивов 98
- Скрипт интерфейса пользователя 14
- Скрипт параметров 14
- Скрипт спецификаций 14
- Сложные команды и дополнительные возможности 18
- Создание 3D-сетки 89
- Создание призмы общего вида 66
- Создание фрагмента поверхности Кунса 86
- Специальные символы 26
- Список значений 21, 26
- Сплайн-кривая 124
- ~ Безье 126
- Способ представления объектов 157
- Ссылка на двоичные данные 183
- Стек преобразований
- удаление из ~ 31
- Стена 47
- криволинейная ~ 49
  - обобщенная ~ 51
- Стиль текста
- установка ~ 157
- Строка в скрипте 23
- Строки символов 24
- Строковое выражение
- длина ~ 206
  - подстрока 207
  - позиция ~ в другом ~ 206
  - разделение ~ 205
  - создание ~ из числового выражения 202
  - строка форматирования 203
  - ширина ~ 206
- Сфера 34
- Т**
- Текст
- ~ в 2D 127
  - ~ в 3D 94
- Тело 95
- Тело на основе примитивов 100
- Тело-поверхность. 112
- Тип линии 175
- определение ~ 174, 175
  - установка ~ 162
- Типы скриптов 13
- Точка в 3D 96
- ~ с указанием начала текстуры 96
- Точка останова 214
- У**
- Управление отбрасыванием теней 160
- Усеченный конус 35
- Условие 210
- Установка образца штриховки
- ~ в 2D-видах 161
- Установка покрытия 158
- Установка цвета пера 156
- Ц**
- Цикл 209
- Цилиндр 33
- ~ выходящий из другого цилиндра 59
  - ~ выходящий из эллипсоида 60

изогнутый ~ 60

## **Ч**

Чертеж

~ в смете элементов 133

ссылка на ~ в 2D-скрипте 184

Числовое выражение 25

## **Э**

Экспорт значений в файл 222

## **Я**

Язык программирования 13