РАЗРАБОТКА ПРИЛОЖЕНИЙ для Unigraphics НА ЯЗЫКЕ GRIP

Любой и каждый может свести лошадь к водопою.
Но если вы научили ее плавать на спине –
считайте, что добились определенных результатов!
Законы Мерфи

Освоив интерактивные средства моделирования, формирования сборочных узлов, оформления конструкторской документации, почти каждый пользователь в процессе решения практических (порой весьма нетривиальных) задач сталкивается с потребностью дополнить имеющиеся в его распоряжении возможности системы новыми функциями для решения специфических проблем или автоматизации выполнения повторяющихся процедур. Очень часто требуется интегрировать Unigraphics (UG) с различными специализированными приложениями, предназначенными для проведения расчетов. При этом геометрия, сформированная в UG, выступает в качестве исходной информации для проведения анализа или, наоборот, Unigraphics берет на себя функции отображения сгенерированной приложением геометрии в модельном пространстве.

Для решения подобных задач Unigraphics располагает достаточно развитыми возможностями модуля UG/Open API (Application Program Interface), реализованного на принципах открытой архитектуры и открывающего доступ к объектам геометрической модели программным приложениям компаний-разработчиков или программам отдельных пользователей. UG/Open дает возможность программным способом создавать геометрические модели, получать информацию об объектах, формировать сборки, генерировать чертежную документацию и т.д. Практически все функциональные возможности Unigraphics, доступные пользователю при интерактивном взаимодействии с системой, реализуемы посредством функций API. Однако обратное утверждение неверно – существует ряд объектов (например, Custom Objects – Объекты пользователя), формирование которых возможно только программным способом.

Модуль UG/Open включает в себя следующие программные интерфейсы:

UG/Open API (User Function). Интерфейс реализует взаимодействие Unigraphics и программ пользователя, написанных на языке C. Заголовочные файлы (.h) соответствуют требованиям стандарта ANSI C и поддерживают разработку программ с использованием языка C++.

В зависимости от способа построения программа пользователя может выполняться как внешнее (External) или как внутреннее (Internal) приложение. В первом случае программа запускается средствами операционной системы как независимое приложение или как процесс, порожденный Unigraphics. Поскольку внешнее приложение не имеет средств графического вывода, ему доступны функции вывода на печатающие устройства и формирования CGI-файла. Во втором случае программа может быть запущена только из текущей сессии Uni-

graphics, загружается в пространство процесса и может выть остановлена (выгружена) только соответствующими командами API, а все результаты работы программы отображаются в графическом окне Unigraphics.

UG/Open GRIP. GRIP (Graphics Interactive Programming – Язык интерактивного графического программирования) представляет собой интерпретируемый язык программирования, использующий инструкции, во многом сходные с такими некогда популярными языками, как BASIC или FORTRAN. Программы, реализованные на языке GRIP, имеют доступ к внутренним программам UG/Open API. В свою очередь, любая GRIP-программа может быть запущена из приложения UG/Open API.

UG/Open GRIP NC. Специализированное расширение языка GRIP для формирования управляющих программ движения инструмента станков с ЧПУ модуля UG/Manufacturing.

UG/Open MenuScript позволяет пользователям и разработчикам программного обеспечения посредством редактирования текстовых ASCII-файлов изменять меню Unigraphics и создавать собственные меню и панели инструментов, интегрированные с их собственными приложениями. Инструментарий MenuScript доступен также программным способом через функции UG/Open API, описанные в заголовочном файле uf_mb.h.

UG/Open UIStyler позволяет пользователям и разработчикам программного обеспечения формировать диалоговые окна и панели Unigraphics.

Следует отметить, что системой лицензирования Unigraphics разделяются права пользователя как на создание приложений UG/Open, так и на их запуск и исполнение (за соответствующими разъяснениями рекомендуем обратиться в компанию Consistent Software, которая является официальным поставщиком Unigraphics).

Описание работы с модулями UG/Open GRIP NC, UG/Open MenuScript, UG/Open UIStyler выходит за рамки рассматриваемой темы, а вот о написании программ пользователя с использованием языков C и GRIP поговорим подробнее. С чего начать? В чем отличие UG/Open GRIP и UG/Open API, и в каких ситуациях следует применять то или иное средство разработки собственных приложений?

Если нужно быстро, что называется на скорую руку, написать небольшое приложение и при этом не имеет большого значения скорость выполнения программы, вы не располагаете опытом программирования и самым большим вашим достижением была BASIC-программа из нескольких строк, выводящая на экран приветствие "Hello, World!", – начните с разработки GRIP-программы. Для этого не потребуется привлекать какие-либо среды разработки типа Microsoft Visual C++: весь необходимый набор инструментов для компилирования и построения исполняемой GRIP-программы поставляется вместе с системой Unigraphics, причем как сам исходный код приложения, так и способ его создания будут абсолютно одинаковы как для платформы Windows NT/2000, так и для Unix-станций Hewlett-Packard, IBM, Sun, SGI и т.д.

Впрочем, следует помнить, что GRIP – интерпретируемый язык. Программы, написанные на нем, выполняются достаточно медленно и малопригодны для обработки больших объемов данных, реализации итерационных расчетных процессов или для моделирования методом Монте-Карло, когда для получения достоверной картины имитируемого процесса требуется произвести огромное количество вычислительных операций. GRIP не располагает процедурами высвобождения используемой памяти, поэтому после запуска программы итерационные процессы постепенно снижают скорость работы.

В отличие от GRIP-программ, приложения, реализованные на языках C или C++, подключаются к Unigraphics в виде динамически подгружаемых библиотек DLL (Dynamic Link Library), выполняются очень быстро и в процессе разработки предоставляют пользователю весь потенциал языка C для управления памятью и ресурсами. Конечно, для создания приложений с использованием UG/Open API требуются опыт написания программ на языке C и навыки работы в среде разработки программ Microsoft Visual C++. При создании серьезных приложений работу по реализации программы разумно поручить квалифицированному программисту, а постановку задачи – инженеру, хорошо понимающему суть решаемой проблемы. Достаточно же простые приложения сможет написать любой программист, когда-либо создававший программы, подобные той, что выводит на экран пресловутое "Hello, World!".

Для разработки приложений UG/Open++ на рабочих станциях под управлением Unix потребуется наличие соответствующих платформе компиляторов:

Платформа	Компилятор
IBM/AIX	Не поддерживается
Hewlett–Packard (HP)	aCC (A.01.12) или aCC (A.01.18)
Digital UNIX (DUX)	Не поддерживается
Silicon Graphics (SGI)	CC (7.2.1)
Sun	CC (4.2)
Windows/NT	Microsoft Visual C++ 6.0 (Service Pack 3)

Здесь нет ни необходимости, ни возможности детально описывать функции, используемые при разработке программного приложения к системе: в руководстве пользователя Unigraphics этот раздел занимает не одну сотню страниц. По собственному опыту могу утверждать, что наиболее трудным в разработке самой первой, самой простой программы является понимание правил компилирования и построения исполняемого модуля, подключения приложения к Unigraphics и его запуска. Именно эти вопросы мы и постараемся рассмотреть наиболее подробно.

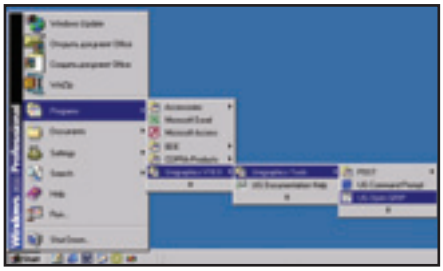
GRADE ± среда разработки GRIP-приложений

Для создания программы, ее компиляции и формирования исполняемого модуля пользователю Unigraphics предлагается GRIP Advanced Development Environment (GRADE) – интегрированная среда разработки. Искушенные программисты могут посетовать на небогатый, на первый взгляд, набор функций этой программной оболочки, однако, независимо от типа рабочей станции (PC или Unix-машина) и

операционной системы (Windows, IRIX, Solaris и т.д.), это окружение будет выглядеть абсолютно одинаково, а исходный код GRIP-программ, созданный на одной платформе, после перекомпиляции работоспособен на любой другой.

Следует отметить, что в системе Unigraphics отдельно предоставляется лицензия как на средства разработки (GRIP Development) GRIP-приложений, так и на запуск (GRIP Execute) исполняемых программ пользователем. Поэтому, прежде чем приступить к созданию собственной GRIP-программы, узнайте у вашего поставщика Unigraphics, располагаете ли вы лицензией на разработку и запуск GRIP-программ.

Запустить программную оболочку GRADE можно из меню рабочего стола Windows: ***Start → Programs → Unigraphics v18.0 → Unigraphics Tools → UG Open GRIP***.



Запуск программной оболочки GRADE

Программная оболочка GRADE после запуска

Рассмотрим подробнее назначение некоторых из десяти пунктов меню GRADE.

Edit – создание нового файла с исходным текстом GRIP-программы, вызов на редактирование существующего файла. В соответствии с принятыми обозначениями для типов файлов Unigraphics файлы с исходным кодом GRIP-программ имеют расширение .grs, а для их редактирования в операционной системе Windows NT/2000 используется простейший текстовый редактор Notepad. Изменить тип текстового редактора можно с помощью восьмого пункта меню, **Change ediTor**, при этом путь к исполняемому файлу нового текстового редактора должен быть включен в системную переменную Windows PATH.

Compile – компилирование исходного текста программы или подпрограмм на языке GRIP и получение объектного кода. Файлы, получаемые в результате компиляции, имеют расширение .gri. В процессе компиляции на экран выводятся содержимое исходного файла и сообщения о возможных ошибках в тексте программы. Управлять количеством информации, выводимой на дисплей, возможно с помощью седьмого пункта меню **Compile listing**. Шестой пункт меню (**Send output to**) определяет устройство, на которое выводится информация: это может быть экран компьютера, принтер или текстовый файл. Если вы не располагаете лицензией на разработку GRIP-программ, при попытке произвести компиляцию созданного приложения оболочка GRADE выдаст соответствующее сообщение.

Сообщение об отсутствии лицензии на разработку GRIP-программ

Link – формирование исполняемого файла GRIP-программы, которая может состоять из одного модуля или включать несколько подпрограмм. Исполняемые файлы, готовые к запуску в сеансе Unigraphics, имеют расширение .grx.

Change directory – этот пункт меню изменяет имя текущей папки, в которой размещены файлы с исходными текстами и в которой будут сохраняться результаты компилирования программ и исполняемые файлы. Соответственно, пункт меню **List directory** позволяет просмотреть содержимое текущей папки с применением фильтра по типу файлов (.grs, .gri, .grx).

Turn menu on/off – для пользователей, достаточно набивших руку в использовании программной оболочки GRADE, может оказаться полезной отмена вывода перечня пунктов меню на экран с сохранением только строки ввода команд.

Любая GRIP-программа должна содержать три обязательные части: объявления или определения, инструкции и завершение программы.

Объявления производятся при помощи четырех ключевых слов (ENTITY, STRING, NUMBER, DATA), которые определяют переменные и их исходные значения для использования в инструкциях программы. Эти объявления резервируют память под числовые значения, объекты, строковые переменные. Например, для определения строковой переменной длиной 30 символов необходимо использовать ключевое слово STRING следующим образом:

```
ΣΤΡΙΝΣ/ΣΤΡ(30)
```

Инструкции GRIP-программы записываются с помощью ключевых слов языка (например: LINE, CIRCLE) и чаще всего используются для создания и манипулирования графическими объектами, вывода сообщений, управления файлами и т.п.

Следующий пример демонстрирует построение четырех линий в форме прямоугольника. Программа содержит объявления переменных (LN1, LN2, LN3, LN4), которым будут соответствовать создаваемые линии и инструкции для их построения. Эта программа создает один и тот же прямоугольник независимо от того, сколько раз будет запущена, так как числовые значения координат вершин жестко определены в коде программы. Позже, при создании программы построения параллелограмма, мы введем дополнительное диалоговое окно для ввода числовых параметров геометрической фигуры.

```
ENTITΨ/ΛN1,ΛN2,ΛN3,ΛN4
ΛN1=ΛINE/0,0,0,2,0,0
ΛN2=ΛINE/2,0,0,2,2,0
ΛN3=ΛINE/2,2,0,0,2,0
ΛN4=ΛINE/0,2,0,0,0,0
```

Для корректного завершения каждая GRIP-программа должна заканчиваться инструкцией HALT.

```
ENTITΨ/ΛN1,ΛN2,ΛN3,ΛN4
ΛN1=ΛINE/0,0,0,2,0,0
ΛN2=ΛINE/2,0,0,2,2,0
```

```
AN3=AINE/2,2,0,0,2,0
AN4=AINE/0,2,0,0,0,0
HALT
```

Любая последовательность символов в пределах строки после $$ игнорируется компилятором и может использоваться для комментирования и документирования программы.

По мере необходимости мы познакомимся и с другими инструкциями языка GRIP, которые потребуются для написания нашего приложения. Для более детального знакомства с языком GRIP рекомендуем обратиться к соответствующему разделу документации Unigraphics.

Создаем первую GRIP-программу

В качестве первого упражнения предлагаю написать небольшую программу на языке GRIP, выполняющую построение параллелограмма по введенным значениям длины основания, высоты, угла при вершине и угла поворота параллелограмма. Разумеется, эту геометрическую фигуру можно построить, используя имеющиеся интерактивные средства моделирования, но применение средств UG/Open GRIP даже при решении такой простой задачи дает заметный выигрыш во времени и сокращает количество интерактивных действий.

Ниже приведен полный исходный текст программы PLOGRAM.GRS:

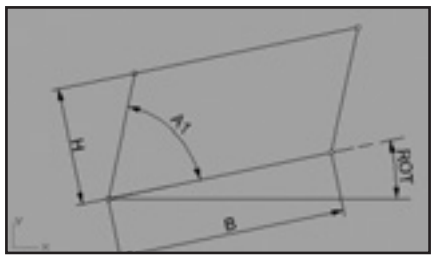


Схема построения параллелограмма

```
$$ Программа: PLOGRAM.GRS
$$
$$ Описание: Программа построения па-
раллелограмма в указанной точке
$$          по длине основания, высоте, углу
при вершине
$$
$$ История разработки: 11-мая-2002
$$ Требуемые подпрограммы: (нет)
$$
$$ Автор : Ю. Чигишев
$$ Компания: Consistent Software
$$ Адрес : Москва, Токмаков пер., 11
$$ Телефон : (095) 913-2222
$$ e-mail : jura@csoft.ru
$$
$$ Операционная система: Windows 2000,
   Unigraphics v18.0
$$

NUMBER/A1,B,H,RESP,ROT,X,Y,Z
ENTITY/L(9),P(5)
DATA/A1,100.0,B,120.0,H,50.0,ROT,20.0

BEG:GPOS/'Укажите точку вставки', X, Y, Z,
RESP
JUMP/BEG:,ENDD:,,,,,RESP

PAR:PARAM/'Введите значения','Длина ос-
нования',B,  $ Ввод параметров
'Высота',H,'Угол при вершине',A1,'Угол по-
ворота',ROT,RESP
   IF/A1*B*H,,,TST:
   MESSG/'Недопустимые параметры'
   JUMP/PAR:

TST:IF/90-ROT, ,NXT:,NXT:  $$ Test Rot Angle
MESSG/'Угол поворота слишком велик'
JUMP/PAR:

NXT:DRAW/OFF          $$ Запрет отрисовки
P(1)=POINT/X,Y,Z
L(1)=LINE/P(1),ATANGL,A1+ROT
   P(2)=POINT/P(1),POLAR,B,ROT
   L(2)=LINE/P(1),P(2)
   L(3)=LINE/PARLEL,L(2),5,H
   P(3)=POINT/INTOF,L(1),L(3)   $$ Опреде-
   ление точки пересечения двух линий
   L(4)=LINE/P(2),PARLEL,L(1)
   P(4)=POINT/INTOF,L(3),L(4)
   DELETE/L(1..4)              $$ Удаление
   вспомогательных линий
   L(5)=LINE/P(1),P(2)         $$ Отрисов-
   ка линий по 4-м точкам
   L(6)=LINE/P(2),P(4)
   L(7)=LINE/P(4),P(3)
   L(8)=LINE/P(3),P(1)
   DRAW/L(5..8)                $$ Отри-
   совка параллелограмма
   DELETE/P(1..4)              $$ Удаление
   вспомогательных точек
   JUMP/BEG:
ENDD:HALT
```

Рассмотрим более подробно алгоритм работы программы и операторы, выполняющие различные действия.

```
NUMBER/A1,B,H,RESP,ROT,X,Y,Z
```

Следующие за ключевым словом NUMBER имена переменных соответствуют определенным числовым значениям, причем безразлично, является ли это число целым, вещественным, одинарной или двойной точности. В зависимости от присваиваемого значения переменная будет соответствовать тому или иному типу числовых данных. В нашем примере переменной A1 соответствует значение угла при вершине параллелограмма, B – длина основания, H – высота параллелограмма, ROT – угол поворота параллелограмма, переменные X, Y, Z будут использованы в промежуточных вычислениях, а переменная RESP – целочисленная величина, которой будут присваиваться коды завершения операций.

```
ENTITY/L(9),P(5)
```

Ключевое слово **ENTITY** объявляет переменные, которые будут соответствовать геометрическим объектам модельного пространства Unigraphics. Так же как и в случае с числовыми данными, объекту типа **ENTITY** может соответствовать точка, линия, кривая, поверхность, твердотельная модель. В нашей программе объявлены девять объектов L (им будут соответствовать отрезки прямых линий) и пять объектов P, которые определяют точки, участвующие во вспомогательных построениях.

Переменные типа **NUMBER** или **ENTITY** могут образовывать массивы, однако размерность их не может превышать 3.

Например, объявление **NUMBER**/NUM(3) определяет три числовых значения, которые образуют одномерный массив, и числовые значения могут быть присвоены элементам массива NUM(1), NUM(2), NUM(3).

Строковые переменные объявляются оператором **STRING**.

Еще одно небольшое замечание: интерпретатором языка GRIP все необъявленные переменные трактуются как "простые" (simple), и им могут быть присвоены только числовые значения. При попытке назначить соответствие необъявленной переменной какому-либо геометрическому объекту последует сообщение об ошибке в процессе компиляции. Пример программы (он не имеет отношения к нашей

задаче построения параллелограмма) приведен ниже:

```
NUMBER/NUM(3)
SIZE=.25
NUM(1)=1.0625
NUM(2)=2.25
NUM(3)=1
HALT
```

Переменная SIZE в этом примере не объявляется – ей просто присваивается числовое значение.

```
DATA/A1,100.0,B,120.0,H,50.0,ROT,20.0
```

Первоначальные значения переменным, которые объявлены как **NUMBER** или **STRING**, могут быть присвоены оператором **DATA**. В нашем примере всем геометрическим размерам будущего прямоугольника назначаются некие величины – несмотря на то, что чуть позже мы предложим пользователю ввести эти значения в соответствующие поля диалогового окна. Если в нашем случае не произвести предварительную инициализацию переменных, диалоговое окно будет содержать случайные значения переменных. Кроме того, оператором DATA очень удобно производить начальные присвоения массивов.

Переменные объявлены, начальные значения присвоены – приступаем к реализации процесса построения параллелограмма. Условимся, что все наши построения происходят в плоскости X-Y текущей системы координат, и прежде всего необходимо определить геометрическое положение вершины параллелограмма. Для этого можно было бы предложить пользователю диалоговое окно ввода координат X и Y для вершины, но есть способ лучше. Язык GRIP предлагает воспользоваться вызовом стандартной процедуры Unigraphics для задания координат точки всеми доступными в интерактивном режиме способами (диалоговое окно *Point Constructor*). Вызов осуществляется при помощи оператора GPOS, описание которого приведено ниже:

```
BEG:GPOS/'Укажите точку вставки', X, Y, Z,
RESP
```

В нашем примере непосредственно оператору GPOS предшествует метка BEG – и прежде чем продолжить описание оператора GPOS, скажем несколько слов об использовании меток в языке GRIP. Язык GRIP не обладает столь же мощными средствами структурирования программы, как, например, C или C++, и для организации условных и безусловных переходов внутри программы очень широко задействован механизм перехода к строке, обозначенной какой-либо меткой в зависимости от выполнения ряда условий. Меткой может быть любая буквенно-цифровая комбинация из шести символов, но в качестве первого символа обязательно должна применяться буква. У многих программистов считается дурным тоном использование инструкций безусловного перехода вида **goto** (в языке GRIP эта инструкция имеет вид **JUMP**/метка:), но в языке GRIP этот прием достаточно часто применим, он встретится и в нашей простейшей программе. Наверное, этого способа передачи управления в программе можно было бы избежать, но иногда для простоты реализации можно воспользоваться и таким методом.

Итак, вернемся к строке, помеченной меткой BEG, и продолжим знакомство с оператором GPOS. За ключевым словом GPOS после символа "/" следует ряд обязательных параметров, первый из которых – заключенное в кавычки сообщение, выводимое в строке подсказки (строка состояния) графического окна Unigraphics в момент активизации диалога *Point Constructor*. В нашем случае в качестве подсказки пользователь увидит сообщение "*Укажите точку вставки*" и далее – подсказки уже от диалога *Point Constructor* для определения координат точки. Мной использована строка на русском языке, и при работе на платформе Windows NT/Windows 2000 с поддержкой русского языка в строке подсказки Unigraphics будет выведено сообщение на русском языке. В то же время попытка скомпилировать на Unix-станциях программу, включающую кириллические символы, может закончиться неудачей (не берусь утверждать это категорично: многие разновидности операционной системы Unix поддерживают русский язык). Поэтому при всем уважении к родному языку рекомендую использовать для заголовков диалоговых окон, сообщений, подсказок строки на английском. Кроме того, руководство по языку GRIP настоятельно не рекомендует использовать символ "*" в заголовках, сообщениях и т.п.

Категорически не рекомендуется использовать в тексте подсказки символ "*". Для корректного выполнения оператора **GPOS** требуется наличие активной модели (Part) – невыполнение этого условия приведет к сообщению об ошибке.

Далее в операторе GPOS следуют ранее декларированные переменные X, Y, Z, которым будут присвоены числовые значения координат, определенных пользователем при помощи диалога *Point Constructor*. Обратите внимание: мы строим плоскую геометрическую фигуру, но координаты вершины определяются в трехмерном пространстве.

RESP	Значение
Кнопка *Back*	1
Кнопка *Cancel*	2
Кнопка *OK*	3
Не используется	4
Координаты определены	5

Последний параметр – целочисленная переменная RESP, которой в зависимости от действий пользователя при завершении работы с диалогом *Point Constructor* присваивается определенное значение (см. таблицу). Например, если после ввода координат точки пользователь нажал кнопку *OK*, переменная RESP приобретает значение 3. Если же пользователь в качестве привязки выбрал, например, концевую точку существующей линии, – диалог *Point Constructor* завершит работу с присвоением переменной RESP значения 5.

Следующий оператор условного перехода в нашей программе в зависимости от значения переменной RESP направит выполнение программы по тому или иному пути. Рассмотрим его подробнее.

JUMP/BEG:,ENDD:,,,,,RESP

В первую очередь обратите внимание на последний параметр этого оператора – целочисленную переменную RESP. Значение этой переменной определяет порядковый номер метки, которой будет передано дальнейшее управление программой. В нашей программе переменная RESP в результате выполнения предыдущего оператора GPOS может принимать целые значения от 1 до 5 – соответственно в операторе JUMP перед параметром RESP следует указать пять меток, которым и будет передаваться управление программой в зависимости от значения переменной RESP. Например, при нажатии кнопки *Back* в диалоге *Point Constructor* переменная RESP принимает значение 1, и оператор JUMP передаст управление программой строке с меткой BEG:, которая еще раз предложит пользователю определить координаты вершины параллелограмма. Напротив, при нажатии кнопки *Cancel* (RESP = 2) завершение всей программы будет прервано, так как вторая метка в нашем операторе JUMP отправляет программу на последний оператор HALT, который завершает работу любой GRIP-программы.

Если нам безразличны остальные значения переменной RESP, соответствующие этим значениям метки можно не указывать, заменив парой запятых, – при этом управление передается оператору, следующему за оператором JUMP. К тому же результату приведет нулевое значение переменной RESP. В любом случае общее количество параметров должно соответствовать возможным принимаемым значениям последнего параметра оператора JUMP.

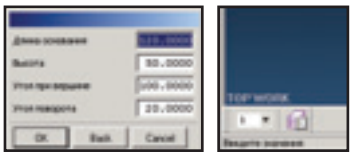
По правилам языка GRIP оператор условного перехода может содержать до 43 меток, которым может быть передано управление. Этого количества более чем достаточно, однако рекомендую вам использовать не более пяти меток в одном операторе JUMP.

В качестве переменной RESP в операторе JUMP может выступать любое вычисляемое выражение (например X+Z); от результата вычисления берется целая часть, и управление передается метке с соответствующим порядковым номером в строке оператора JUMP. В то же время пользоваться таким управлением следует достаточно осторожно.

Мы достаточно подробно рассмотрели оператор условного перехода – в языке GRIP он применяется несколько своеобразно. Переходим к следующему оператору в нашей программе:

PAR:**PARAM**/'Введите значения', 'Длина основания', B, $ Ввод параметров
'Высота', H, 'Угол при вершине', A1,'Угол поворота', ROT, RESP

Сам оператор имеет ключевое слово **PARAM**. Назначение метки PAR: поясним чуть позже. При выполнении этого оператора GRIP-программа предлагает пользователю ввести определенные параметры в диалоговом окне, которое в нашем случае будет выглядеть так, как это показано на рисунке:



Первый из параметров оператора **PARAM** – сообщение длиной до 40 символов, выводимое в строке состояния графического окна Unigraphics; далее парами следуют названия предлагаемых полей ввода числовых значений и имена переменных, которым эти значения будут присвоены. GRIP позволяет создавать диалоговые окна, содержащие до 14 полей ввода, названия которых могут содержать до 15 символов. В общем случае оператор **PARAM** может оказаться достаточно длинным, и для его записи в несколько строк можно воспользоваться символом "$".

Обратите внимание: начальные значения переменных в полях ввода соответствуют присвоенным оператором **DATA**. В рассматриваемом примере все числовые значения имеют вещественный тип и в поле ввода они отображаются с десятичной точкой и дробной частью. Оператор PARAM позволяет определить вводимый тип данных как целое при помощи специального ключевого слова INT, в этом случае в поле ввода число будет отображаться без десятичного разделителя. Пример записи оператора в этом случае выглядит следующим образом:

PARAM/'Введите число повторений', 'Количество', INT, B, RESP

Категорически не рекомендуется использовать в названиях полей ввода и строке подсказки символ "*". Для корректного выполнения оператора **PARAM** требуется наличие активной модели (Part), невыполнение этого условия приведет к сообщению об ошибке.

RESP	Значение
Back	1
Cancel	2
OK	3

Выводимое диалоговое окно имеет три кнопки: *Back*, *Cancel*, *OK* – и в зависимости от того, какую кнопку нажмет пользователь, переменной RESP присваиваются различные значения (см. таблицу). Переменная RESP может принимать и значение RESP = 4, но рассказ о том, в каких случаях это происходит, выходит за рамки рассматриваемого примера (для более детального изучения работы оператора **PARAM** советуем обратиться к документации по UG/Open). В данном случае нас не интересует, какую именно кнопку нажмет пользователь, изменит или не изменит он значения геометрических параметров – далее мы будем проверять именно корректность введенных данных. Естественно, что длина стороны параллелограмма, его высота, угол при вершине не могут иметь нулевые или отрицательные значения, поэтому используем оператор условного перехода другого типа, с классическим ключевым словом IF:

IF/A1*B*H,,,TST:

В общем виде оператор IF имеет следующую форму записи:

IF/[Выражение], label1, label2, label3

В зависимости от значения вычисляемого выражения управление программой будет передано оператору с соответствующей меткой. Если результат вычисления отрицательный – управление передается оператору с меткой label1, результат равен нулю – выполняется оператор с меткой label2, а в случае положительного значения результата вычисления выражения происходит переход к строке с меткой label3. Если какая-либо из трех меток в записи оператора отсутствует, управление передается оператору, следующему за оператором **IF**.

В нашей программе в качестве вычисляемого выражения оператора IF представлено произведение переменных A1, B, H (угол при вершине, длина стороны и высота соответственно), и в случае ввода отрицательных или нулевых значений для какого-либо геометрического параметра результат вычисления выражения будет нулевым или отрицательным. А так как первая и вторая метка в нашем операторе IF отсутствуют, происходит переход к следующему оператору, который выводит сообщение о неверно введенных значениях:

MESSG/'Недопустимые параметры'

Сообщения в GRIP выводятся в модальное окно – это означает, что программа приостанавливает свое выполнение до закрытия окна. В нашем примере после закрытия окна следует безусловный переход к оператору, помеченному меткой PAR: (пользователю предлагается исправить неверно введенные параметры).

Если же введенные параметры – положительные величины, переходим к оператору с меткой TST:. Здесь происходит проверка введенного значения угла поворота. Предлагаю читателю самостоятельно разобрать логику работы этого участка программы. Мы же сразу перейдем к оператору с меткой NXT:

NXT:**DRAW**/OFF

Действие оператора DRAW с параметром OFF очень простое – начиная с этого момента запрещен вывод любой геометрической информации в графическое окно Unigraphics. Такой запрет нам потребовался для того, чтобы скрыть промежуточные, вспомогательные построения.

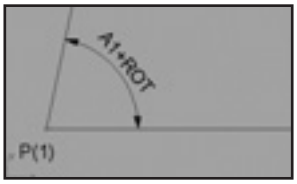
Далее начинается самое интересное: построение геометрических примитивов – вспомогательных точек, линий!

P(1)=**POINT**/X,Y,Z

В разделе объявлений нашей программы был объявлен массив из пяти точек. Приведенный выше оператор производит построение первой из них с координатами X, Y, Z, которые были определены при помощи оператора **GPOS**. Это самый простой способ построения точки с известными координатами. Средства языка GRIP позволяют программным способом реализовать практически все способы интерактивного построения точек, некоторые из них представлены в таблице.

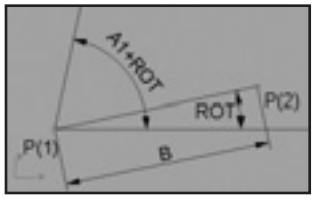
L(1)=**LINE**/P(1),ATANGL,A1+ROT

От первой построенной точки – вершины будущего параллелограмма, проводим линию под углом, величина которого равна сумме угла при вершине и угла поворота параллелограмма. Обратите внимание: длина этой линии нами не определяется, в данном случае объект L(1) – луч, имеющий начало в точке P(1) и уходящий в бесконечность (конечно же, в пределах модельного пространства Unigraphics!). Это одна из причин, по которой полезно временно отключить вывод геометрических объектов на экран.



P(2)=**POINT**/P(1),POLAR,B,ROT

Вторую точку P(2) построим, применяя смещение от вершины в полярных координатах на угол поворота параллелограмма, и на расстояние, равное длине основания.



L(2)=**LINE**/P(1),P(2)

Строим линию L(2), соединяющую точки P(1) и P(2).

L(3)=**LINE**/PARLEL,L(2),5,H

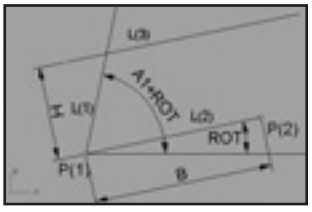
Проводим линию L(3) параллельно линии L(2) на расстоянии H (высота параллелограмма). Обратите внимание на третий параметр оператора с ключевым словом LINE – целое число 5. Дело в том что при построении линии, параллельной заданной на указанном расстоянии,

Оператор	Описание
POINT/X,Y,Z	Построение точки по известным координатам X, Y, Z.
POINT/CENTER,circle	Построение точки в центре существующей окружности circle.
POINT/INTOF,obj1,obj2	Построение точки пересечения двух объектов; в качестве первого объекта obj1 должна выступать кривая, а второй объект obj2 может быть кривой, плоскостью, гранью твердого тела. Здесь приведено упрощенное описание этого оператора, более подробно все параметры изложены в документации UG/Open.
POINT/point,VECT,line,dist	Построение точки на заданном расстоянии dist от существующей точки point в направлении вектора, параллельного существующей линии line.
POINT/circle,ATANGL,angle	Построение точки по заданному угловому положению angle на окружности circle.
POINT/point,DELTA,dx,dy,dz	Построение точки в виде смещения от существующей точки на заданные расстояния по X, Y, Z.
POINT/ENDOF,"PMOD3",obj	Построение концевой точки линии, кривой.
POINT/point,POLAR,dist,angle	Построение точки в виде смещения от существующей точки в полярных координатах (по углу и расстоянию).

необходимо указать, с какой стороны провести эту линию. Чтобы не приостанавливать выполнение программы и не спрашивать пользователя о том, с какой стороны провести линию (в принципе это возможно, но мы же хотим построить параллелограмм автоматически!), язык GRIP предлагает использование так называемых пространственных модификаторов и соответствующих им числовых кодов:

XSMALL – 1 – PMOD2/PMOD3
YSMALL – 2 – PMOD2/PMOD3
ZSMALL – 3 – PMOD3
XLARGE – 4 – PMOD2/PMOD3
YLARGE – 5 – PMOD2/PMOD3
ZLARGE – 6 – PMOD3

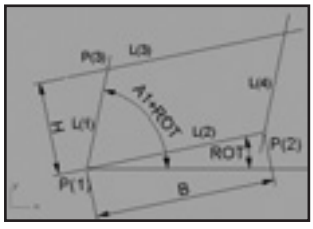
В нашем примере мы использовали модификатор YLARGE (код



5) – проще говоря, предложили провести параллельную линию в сторону бóльших координат Y относительно линии L(2).

P(3)=**POINT**/INTOF,L(1),L(3)
L(4)=**LINE**/P(2),PARLEL,L(1)

Находим точку пересечения линий L(1) и L(3) – точку P(3) и проводим через точку P(2) линию L(4), параллельную линии L(1).



P(4)=**POINT**/INTOF,L(3),L(4)

DELETE/L(1..4)

L(5)=**LINE**/P(1),P(2)
L(6)=**LINE**/P(2),P(4)
L(7)=**LINE**/P(4),P(3)
L(8)=**LINE**/P(3),P(1)

Далее находим точку пересечения линий L(3) и L(4) – точку P(4), удаляем вспомогательные построения – линии L(1), L(2), L(3), L(4) и строим "чистовые" линии, то есть стороны нашего параллелограмма – линии L(5), L(6), L(7), L(8). Обратите внимание на способ записи в операторе DELETE списка объектов: указаны первый и последний элементы списка, которые разделены *двумя* точками.

DRAW/L(5..8)
DELETE/P(1..4)
JUMP/BEG:

Мы на финишной прямой – наконец-то выводим изображения построенных линий, удаляем вспомогательные точки и любезно предлагаем пользователю построить еще один параллелограмм, осуществляя переход к оператору определения вершины **GPOS**.

Конечно же, пока мы только шаг за шагом "проиграли" алгоритм выполнения программы, познакомились с некоторыми операторами языка GRIP и получили законченный исходный код программы.

(Продолжение следует)

Юрий Чигишев
Consistent Software
Тел.: (095) 913-2222
E-mail: jura@csoft.ru