ГЛАВА

35

Программирование на AutoLISP

В этой главе...

Создание переменных

Одним из важнейших элементов большинства языков программирования являются переменные. *Переменная* — это символьное наименование, которым можно оперировать в данной программе. Важным фактором использования переменных является возможность присвоения им значений. Рассмотрим пример, в котором переменной `radius` присваивается значение 3.

```
(setq radius 3)
3
```

Эту операцию можно выполнить в окне Console Visual LISP. Если вы хотите использовать эту переменную в командной строке AutoCAD, перед ней следует поставить восклицательный знак (!), например:

```
Command:!radius
3
```

Восклицательный знак перед переменной заставляет программу вычислить (оценить) значение, хранящееся в переменной, и вывести его в командную строку. При работе с переменными в окне Console восклицательный знак перед именем переменной не нужен, поскольку предполагается, что любая вводимая в этом окне информация представляет собой выражение AutoLISP.

Взаимосвязь между строковыми константами и переменными так же проста, как между числовыми константами и переменными.

```
(setq name "Robin")
"Robin"
```

Можно также использовать вложенные AutoLISP-выражения — включать их одно в другое:

```
(setq radius (+ 2 1))
3
```

Как было описано в предыдущей главе, AutoLISP сначала вычисляет значение выражения во внутренних скобках `(+ 2 1)`, а затем присваивает результат переменной `radius`.

Пошаговая инструкция

Использование переменных AutoLISP в AutoCAD

1. Откройте новый чертеж, используя опцию Start from Scratch (Без шаблона).
2. Введите **`(setq radius (+ 2 1))`**↵. AutoLISP возвратит `3`.
3. Запустите команду **`CIRCLE`** (КРУГ). Задайте центр окружности. В ответ на приглашение `Specify radius of circle or [Diameter]:` (Задайте радиус круга или [Диаметр]:) введите **`!radius`**↵. AutoCAD начертит окружность радиусом 3 единицы.
4. Введите **`(setq color "green")`**↵. AutoLISP возвратит `"green"` (зеленый).
5. Введите **`color`**↵. В ответ на приглашение `Enter default object color <BYLAYER>:` (Введите цвет объекта по умолчанию <ПОСЛОЮ>:) введите **`!color`**↵.
6. Создайте новую окружность. Она будет вычерчена на экране зеленым цветом, так как именно этот цвет назначен текущим.
7. Сохраните чертеж в папке `AutoCAD Bible` под именем `ab35-1.dwg`.

Работа с командами AutoCAD

Доступность команд AutoCAD из языка AutoLISP служит мощным средством автоматизации часто выполняемых функций. Комбинируя команды AutoCAD с переменными AutoLISP, как описано в предыдущем разделе, можно достичь высокой степени гибкости программирования.

Доступ к командам AutoCAD

В предыдущей главе уже упоминалась функция `COMMAND` при знакомстве с AutoLISP-программой. Эта функция используется в языке AutoLISP для выполнения команд AutoCAD. Она воспринимает все последующие операнды так, будто они вводятся в командную строку в интерактивном режиме. При программировании функции `COMMAND` в AutoLISP нужно в точности дублировать то, что вводилось бы в командную строку. Например, для того чтобы начертить отрезок, необходимо следовать рекомендациям, изложенным в следующей таблице. В первой колонке представлены действия пользователя при создании отрезка в AutoCAD, а во второй показано, как те же действия описать в AutoLISP-программе.

Действия пользователя в AutoCAD	*Соответствующие программные конструкции AutoLISP*
Ввести **`line`** (ОТРЕЗОК) в командную строку	`"line"` (или `"_line"`)
Нажать <Enter>	Используйте пробел, который в AutoLISP соответствует нажатию <Enter> после ввода команды

Действия пользователя в AutoCAD	*Соответствующие программные конструкции AutoLISP*
Определить начальную точку для отрезка	Можно использовать переменную, значения координат в явном виде или запрограммировать паузу для ввода значений пользователем
Определить конечную точку	Можно использовать переменную, значения координат в явном виде или запрограммировать паузу для ввода значений пользователем
Нажать <Enter> для завершения команды **LINE** (ОТРЕЗОК)	Для того чтобы в AutoLISP-программе описать нажатие <Enter> в процессе выполнения команды или при ее окончании, нужно использовать пару двойных кавычек

Например, при использовании переменных `startpt` и `endpt` для начальной и конечной точек отрезка обращение к команде **LINE** из AutoLISP-выражения осуществляется так:

```
(command "_line" startpt endpt "")
```

Создание функций

Использование функций всегда начинается с оператора `DEFUN`. Можно определить три основных типа функций.

- К первому типу относятся функции, определяемые оператором `DEFUN`, именам которых предшествует `C:` (до сих пор вы встречались именно с такими функциями). Это позволяет использовать имя функции в командной строке AutoCAD. Такую функцию можно использовать, как любую команду AutoCAD.
- Определение функции можно формировать и без префикса `C:`. Этот тип функций наиболее удобен, если функция вызывается другими AutoLISP-выражениями. Для обращения к ней из командной строки необходимо заключить имя функции в круглые скобки. Заметим, что в AutoLISP-выражениях можно также использовать функции с префиксом `c:`. Для этого в круглые скобки необходимо заключить имя функции вместе с префиксом, например `(c:circle3)`.
- Третий тип функций — `S::STARTUP`. Если функция определена с именем `S::STARTUP` (обычно именно такое определение вы встретите в файлах `acaddoc.lsp`), она будет автоматически выполнена при инициализации нового чертежа или запуске AutoCAD.

При создании функции `S::STARTUP` необходимо хорошо продумать, в какой файл ее поместить. В главе 34 кратко упоминались различия между файлами `acad.lsp` и `acaddoc.lsp`. Необходимость создания двух файлов автозагрузки в AutoCAD 2000 объясняется тем, что в этой версии появился многооконный интерфейс — MDI, который позволяет открывать несколько чертежей одновременно. Более подробная информация по этому вопросу содержится на врезке *Автоматически загружаемые файлы .lsp*. Использование функции `S::STARTUP` значительно повышает производительность, так как позволяет автоматизировать любые стандартные операции, которые обычно выполняются в начале сеанса работы в AutoCAD или при открытии нового чертежа.

Автоматически загружаемые файлы `.lsp`

AutoCAD автоматически загружает четыре файла AutoLISP. Два из них, `acad2000.lsp` и `acad2000doc.lsp`, появились лишь в AutoCAD 2000. Компания Autodesk рекомендует зарезервировать эти файлы для нужд разработчиков самой системы AutoCAD. Файл

acad2000.lsp загружается один раз при загрузке AutoCAD, а acad2000doc.lsp загружается при открытии каждого нового чертежа.

Два других автоматически загружаемых файла AutoLISP — acad.lsp и acaddoc.lsp — предназначены для пользователей системы. Файл acad.lsp загружается один раз перед началом сеанса работы в AutoCAD, а файл acaddoc.lsp — перед открытием каждого нового чертежа. Это означает, что в эти файлы можно поместить различные программы инициализации: одну для инициализации AutoCAD (в файл acad.lsp), а другую для каждого нового чертежа. Функции S::STARTUP можно поместить в оба файла. Однако следует помнить, что, помещая различные функции в оба файла, вы, по существу, отключаете функцию S::STARTUP, определенную в файле acad.lsp.

Вам необходимо создать оба файла — acad.lsp и acaddoc.lsp. Однажды создав каждую из этих программ AutoLISP и сохранив ее в одном из вышеназванных файлов, вы впоследствии сможете добавить в этот же файл другие программы. AutoCAD будет автоматически загружать эти файлы до тех пор, пока они будут существовать.

Файлы инициализации загружаются в следующем порядке:

1. Acad2000.lsp. Загружается в AutoCAD 2000 первым.
2. Acad.lsp. Этот определенный пользователем файл загружается один раз при загрузке AutoCAD 2000.
3. Acad2000doc.lsp. Системный файл инициализации документа, загружаемый AutoCAD 2000.
4. Acaddoc.lsp. Определенный пользователем файл инициализации документа.

Ниже приведен текст AutoLISP-программы, в которой используются функции DEFUN и COMMAND:

```
(defun c:redline (/ startpt endpt)
  (terpri)
 (setq startpt (getpoint "Выберите начальную точку для
redline:"))
  (terpri)
(setq endpt (getpoint "Выберите конечную точку для redline:"))
  (command "_line" startpt endpt "")
  (command "_chprop" "_last" "" "_color" "red" "")
)
```

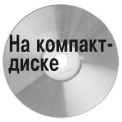
Рассмотрим эту программу подробнее.

- Первая строка программы определяет функцию с именем redline. Так как имени redline предшествует префикс c:, то имя этой функции можно ввести в командную строку. Как и в рассмотренной ранее программе circle3, выражение (/ startpt endpt) означает, что redline имеет две локальные переменные, доступные только в этой программе. Эти переменные используются в следующих двух строках программы.
- Новая инструкция terpri во второй и четвертой строках означает, что AutoCAD выведет в командной строке пустую строку перед тем, как выводить приглашения. В таком виде они лучше воспринимаются пользователем. В противном случае в одной строке может появиться несколько приглашений.
- Считывая в третьей строке выражение во внутренних круглых скобках, AutoCAD получает сведения о координатах начальной точки, введенной пользователем в ответ на приглашение Выберите начальную точку для redline:, и присваивает переменной startpt значение, соответствующее координатам начальной точки. В пятой строке аналогично присваивается значение переменной endpt.

- В шестой строке используется AutoLISP-функция `COMMAND`. В ней выполняется команда **`LINE`**, задаются начальная и конечная точки и используется пустая пара двойных кавычек для того, чтобы отразить в программе нажатие <Enter> после завершения ввода последовательности узловых точек команды **`LINE`**.
- В седьмой строке используется такой же синтаксис для команды **`CHPROP`** (СВОЙСТВА). В ней с помощью опции `Last` (Последний выбор) выбирается только что созданный отрезок, а пустая пара двойных кавычек заканчивает операцию выбора объекта. Далее задается опция `Color` (Цвет) и устанавливается красный цвет. Последняя пустая пара двойных кавычек завершает выполнение команды.
- В восьмой строке программа `redline` завершается закрывающей круглой скобкой.

Для работы с этой программой выполните ряд операций.

1. Откройте Visual LISP и создайте новый файл программы.
2. Наберите текст программы.
3. Сохраните программу в файле `redline.lsp` и поместите ее в папку `Support` AutoCAD или в любую другую папку, созданную для AutoLISP-программ и добавленную в список путей поиска файлов поддержки AutoCAD.
4. Загрузите программу.
5. Перейдите в окно AutoCAD.
6. Введите в командной строке **`redline`**↵.
7. В ответ на приглашения укажите две точки на экране, которые задают начало и конец отрезка. AutoCAD начертит между указанными точками отрезок красного цвета.



Файл `redline.lsp` находится на прилагаемом компакт-диске. Его можно просто скопировать на свой компьютер и поэкспериментировать с ним. Например, вспомнив упражнение с программой `circle3` из главы 34, вы можете создать окружность красного цвета.

Ниже приводится другой пример AutoLISP-программы, которая создает функцию `S::STARTUP`. В ней используется несколько положений, изложенных в этой главе.

```
(defun s::startup ()
   (command "_rectang" "_width" "0.1" "0.0" "10.10")
   (command "_text" "8,1" "0.2" "0" "name")
   (command "_text" "8,0.7" "0.2" "0" "date")
   (command "_text" "8, 0.4" "0.2" "0" "revision")
   (command "_zoom" "_extents")
)
```

Каждый раз, когда открывается новый чертеж, эта программа создает штамп чертежа и рамку (рис. 35.1).

Для использования этой или любой другой созданной вами программы необходимо добавить ее в конец файла `acaddoc.lsp`. Этот файл не входит в комплект поставки AutoCAD 2000, поэтому вам придется создать его самостоятельно.

Прежде чем использовать функцию `S::STARTUP`, необходимо убедиться, что ее еще нет в этом файле. Функцию `S::STARTUP` могут включать в файл автозагрузки некоторые другие установленные вами приложения. Появление второй функции `S::STARTUP` в файле автозагрузки может нарушить работу этих приложений. Поэтому в редакторе *Notepad* выберите Search⇨Find

(Поиск⇨Найти) и введите `s::startup` в поле ввода Find what (Что искать). Щелкните на кнопке Find next (Найти следующий). Если функция `S::STARTUP` найдена, добавьте тело новой функции (без первой строки, которая уже имеется в файле) в конец существующей программы `S::STARTUP` и сохраните файл.

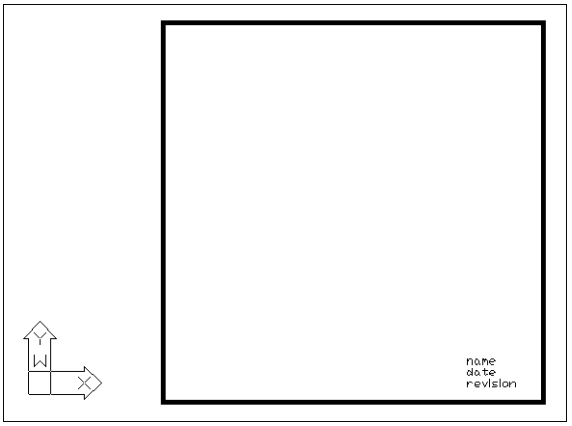


Рис. 35.1. Результат выполнения программы `S::STARTUP`

Создание функций с аргументами

Можно создавать функции с аргументами. *Аргумент* — это значение, которое должно передаваться функции. Функция использует значение аргумента в процессе выполнения.

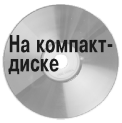
Ранее в этой главе уже упоминалось, что локальные переменные помещаются в круглых скобках после косой черты. Аргументы тоже располагают в круглых скобках, но только до косой черты. Если локальные переменные в функции отсутствуют, косая черта не ставится. Ниже приводится пример функции с одним аргументом.

```
(defun chg2red (selected_object)
...
)
```

Для обращения к этой программе из AutoCAD или из другой AutoLISP-программы используется формат `(chg2red selected_object)`. Здесь имя переменной, которая передается программе `chg2red` в качестве аргумента, находится после имени функции, и все выражение заключается в круглые скобки.

При каждом обращении к функции `chg2red` в какой-либо программе необходимо задавать значение аргумента. В качестве аргумента можно использовать конкретное значение или имя определенной в программе переменной, значение которой вводится пользователем в процессе работы программы.

В следующем упражнении создается функция, которая вызывается в AutoLISP-программе.



Файл `ab35-a.lsp`, используемый в следующем упражнении, содержится на прилагаемом компакт-диске. Если вы еще не поместили этот компакт-диск в дисковод CD-ROM, сделайте это сейчас.

Пошаговая инструкция

Использование функций и команд AutoLISP

1. Откройте новый чертеж, используя опцию Start from Scratch (Без шаблона).
2. С помощью *Windows Explorer* скопируйте файл `ab35-a.lsp` с компакт-диска в папку `Support` или в другую папку, созданную для AutoLISP-файлов. Во втором случае путь к этой папке должен быть добавлен в список путей поиска файлов поддержки. Этот файл показан на рис. 35.2.

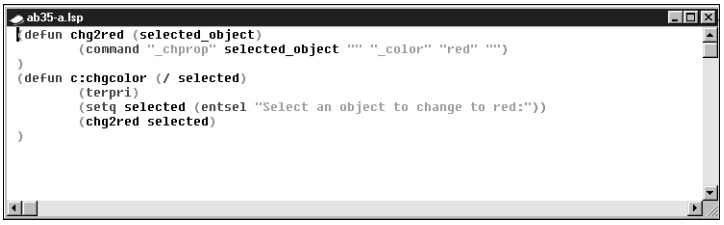
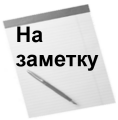


Рис. 35.2. AutoLISP-программа для изменения цвета объекта на красный

3. Выберите команду Tools⇨AutoLISP⇨Visual LISP Editor (Сервис⇨AutoLISP⇨Редактор Visual LISP). В редакторе Visual LISP откройте файл `ab35-a.lsp`. Текст программы появится в окне редактирования, как показано на рис. 35.2.
4. Щелкните на пиктограмме Load active edit window (Загрузить программу из активного окна редактирования).
5. Щелкните на пиктограмме Activate AutoCAD (Активизировать AutoCAD).
6. Начертите любой объект.
7. В командной строке введите **chgcolor**↵.
8. В ответ на приглашение `Select an object to change to red:` (Выберите объект для изменения цвета на красный:) выберите начерченный объект. Его цвет изменится на красный.
9. Не сохраняйте программу и измененный чертеж.

Ниже описана работа программы.

- Эта программа определяет функцию `chg2red`, которой не предшествует префикс `c:`. У нее один аргумент `selected_object`.
- После ввода `chgcolor` AutoLISP фактически выполнил функцию `c:chgcolor`. В последнее выражение этой функции — `(chg2red selected)` — переменная `selected` поступила из предыдущей строки как результат выполнения операции `entsel` (*ENTity SELect*) (Выбор примитива).
- Переменная `selected` — это аргумент, который передается в функцию `chg2red`. Теперь функции известно, каким объектом оперировать.
- Функция `chg2red` использует далее команду **CHPROP** (СВОЙСТВА) для изменения цвета на красный.



Для вызова этой функции из командной строки необходимо ввести **(chg2red** *apr***)**, где *apr* — это имя аргумента (в данном случае в качестве аргумента можно использовать имя примитива). Имена примитивов обсуждаются ниже в этой главе.

Работа с системными переменными

AutoCAD содержит множество системных переменных, предназначенных для управления процессом вычерчивания. К счастью, разработчики AutoLISP обеспечили возможность программистам автоматизировать установку и считывание значений системных переменных AutoCAD.

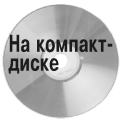
Не следует путать термины *переменные* и *системные переменные*. Переменная — это значение, которое создается для использования в программе, а системная переменная — установка AutoCAD, которая служит для настройки режима работы AutoCAD.

Для установки или считывания системных переменных используются операторы SETVAR и GETVAR, которые могут применяться к любым системным переменным AutoCAD.

- Название оператора SETVAR — сокращение от *SET VARiable* (Установить переменную). Оператор SETVAR используется для изменения значения системной переменной. После имени системной переменной, заключенного в двойные кавычки, следует поместить ее новое значение: `(setvar "cmdecho" 0)`.
- Название оператора GETVAR — сокращение от *GET VARiable* (Получить переменную). Этот оператор позволяет получить значение системной переменной, которое можно присвоить другой переменной. Это часто делается для того, чтобы вернуть системной переменной прежнее значение, которое изменялось в процессе работы программы. Имя системной переменной должно быть заключено в двойные кавычки: `(getvar "cmdecho")`.

Операторы SETVAR и GETVAR могут применяться к любым системным переменным. Ниже приводятся две системные переменные, которые часто изменяются в AutoLISP-программах.

- Во всех AutoLISP-программах, с которыми мы имели дело до сих пор, сообщения команд AutoCAD можно было увидеть в зоне командной строки. Системная переменная **CMDECHO** определяет, будут ли отображаться приглашения и введенные параметры в процессе работы функции COMMAND. По умолчанию режим их отображения включен (переменная имеет значение 1). Если его изменить на 0, то приглашения и введенные параметры не будут выводиться в командную строку, но функционирование AutoLISP-программы будет более наглядным, и процесс несколько ускорится.
- Системная переменная **FILEDIA** включает и выключает вывод на экран диалоговых окон выбора файлов. Выключение этой системной переменной позволяет задавать имена файлов в командной строке.



Файл `ab35-a.lsp`, используемый в следующем упражнении, содержится на прилагаемом компакт-диске. Если вы еще не поместили этот компакт-диск в дисковод CD-ROM, сделайте это сейчас.

Пошаговая инструкция

Использование языка AutoLISP для работы с системными переменными

1. Если при выполнении предыдущих упражнений вы скопировали файл `ab35-a.lsp` в папку файлов поддержки AutoCAD `Support` или в другую папку, созданную для AutoLISP-программ, откройте его. В противном случае скопируйте файл `ab35-a.lsp` с прилагаемого компакт-диска в папку `Support` или в другую папку, путь к которой включен в список путей поиска файлов поддержки.
2. Создайте новый чертеж, используя опцию Start from Scratch (Без шаблона). Выберите команду Tools⇨AutoLISP⇨Visual LISP Editor (Сервис⇨AutoLISP⇨Редактор Visual LISP). Щелкните на кнопке Open (Открыть) и откройте файл `ab35-a.lsp`. Отредактируйте его, чтобы содержимое файла выглядело следующим образом:

```
(defun chg2red (selected_object)
   (command "_chprop" selected_object "" "_color" "red" "")
)
(defun c:chgcolor (/ selected old_cmdecho)
   (setq old_cmdecho (getvar "cmdecho"))
   (setvar "cmdecho" 0)
   (terpri)
   (setq selected (entsel "Выберите объект для изменения
цвета на       красный:"))
   (cng2red selected)
   (setvar "cmdecho" old_cmdecho)
)
```

3. Сохраните файл под именем `ab35-1.lsp` в том же каталоге.
4. Щелкните на кнопке Load active edit window, чтобы загрузить программу.
5. Начертите произвольный объект.
6. В командной строке введите **chgcolor**↵.
7. В ответ на приглашение `Выберите объект для изменения цвета на красный:` выберите объект, начерченный на шаге 5. Команды AutoCAD теперь не выводятся в командную строку. Выбранный объект стал красным, и AutoCAD немедленно отобразил приглашение на ввод следующей команды.
8. Не сохраняйте чертеж.

Ниже описывается работа программы. Надеемся, что вы прочли описание работы предыдущей программы, очень похожей на эту.

- Во-первых, в функцию `chgcolor` добавлена новая переменная `old_cmdecho`.
- В следующей строке этой переменной присваивается текущее значение системной переменной **CMDECHO**. Это значение считывается оператором `GETVAR`.
- Далее системная переменная AutoCAD **CMDECHO** с помощью `SETVAR` устанавливается в 0.
- В процессе отладки программ может потребоваться прочитать сообщения AutoCAD. Поэтому лучше, наверное, восстановить значение переменной **CMDECHO**, которое было до выполнения программы. В связи с этим в последней строке снова используется `SETVAR` для присвоения переменной **CMDECHO** значения переменной `old_cmdecho`, в которой было сохранено исходное значение **CMDECHO**.

В результате этих изменений программа `chgcolor` всегда восстанавливает значение системной переменной.

Работа со списками

Списки — это основные структуры данных, используемые в программировании на языке AutoLISP. Осваивая последующий материал данной главы, вы получите представление о применении списков как при обработке объектов (называемых также примитивами) базы данных AutoCAD, так и в других контекстах AutoLISP. AutoCAD представляет данные об объекте в виде списка, который содержит много других списков меньшего размера. Хотя на первый взгляд такая структура выглядит сложной (а правильнее сказать — непривычной), обработка списков предельно проста и понятна.

Применение списков для задания координат

Список всегда заключается в круглые скобки. Одна из простейших и широко применяемых в AutoCAD списочных структур — набор координат точки, например:

```
(1.0 3.5 2.0)
```

Этот список задает точку с координатами `1.0,3.5,2.0` в прямоугольной системе координат `X,Y,Z`. Так как список представляет собой группу элементов, может потребоваться извлечь из него один или несколько элементов. В табл. 35.5 приведен перечень функций извлечения элементов на примере списка `(1.0,3.5,2.0)`.

Таблица 35.5. Основные функции извлечения элементов списка

Функция	*Результат*	*Описание*
`CAR`	`1.0`	Возвращает первый элемент списка
`CDR`	`(3.5 2.0)`	Возвращает все элементы списка, кроме первого
`CADR`	`3.5`	Возвращает второй элемент списка
`CADDR`	`2.0`	Возвращает третий элемент списка

Для большей гибкости можно использовать функцию `NTH`, которая позволяет получить доступ к произвольному элементу списка и извлекает элемент указанного списка с нужным номером. Для этого ей нужно передать два аргумента: первый задает номер элемента списка (заметим, что нумерация элементов списка начинается с `0`), а второй указывает на сам список.

Как правило, имя списка — переменная, значение которой присваивается оператором `setq`,, например:

```
(setq corner (list 1.0 3.5 2.0))
(nth 0 corner)
```

В этом примере функция `(nth 0 corner)` возвращает значение `1.0`, так как `1.0` является первым элементом списка `corner`.

Список создается функцией `LIST`. Если элементы списка являются константами (а не переменными), для его формирования можно использовать функцию `QUOTE`. Для ускоренного вызова функции `QUOTE` можно использовать одиночную кавычку (') (она задается той же клавишей, что и апостроф). Следующие две функции эквивалентны:

```
(setq corner (list 1.0 3.5 2.0))
(setq corner '(1.0 3.5 2.0))
```

Многие другие AutoLISP-функции, предназначенные для извлечения элементов из списка, описаны в приложении *Appendix A* руководства пользователя Visual LISP *Visual LISP Devel-*

oper's Guide. Сначала обратитесь к разделу *Basic Functions* (Базовые функции), а затем *List Manipulation Functions* (Функции манипулирования списками). Ограниченный объем данной книги не позволяет представить на ее страницах все многообразие функций AutoLISP.

Создание точечных пар

Точечная пара — это специальный тип списка, который содержит только два элемента. Некоторые AutoLISP-функции не допускают точечных пар в качестве аргумента, хотя они и используются для представления объектов в базе данных AutoCAD. Точечную пару формирует функция `CONS`, например

```
(cons 2.1 4.5)
```

возвращает `(2.1 . 4.5)`. Список такого типа называется *точечная пара*, поскольку содержит два элемента, разделенных точкой.

Пошаговая инструкция

Работа со списками в AutoLISP

1. Создайте новый чертеж, используя опцию Start from Scratch.
2. Выберите команду Tools⇨AutoLISP⇨Visual LISP Editor, чтобы открыть редактор Visual LISP.
3. В окне Console введите **`(setq endpt '(3.5 2.0 1.4))`**↵. AutoLISP возвратит `(3.5 2.0 1.4)`.
4. Вернитесь в окно Console. Введите **`(car endpt)`**↵. AutoLISP возвратит `3.5`.
5. Введите **`(cadr endpt)`**↵. AutoLISP возвратит `2.0`.
6. Введите **`(cdr endpt)`**↵. AutoLISP возвратит `(2.0 1.4)`.
7. Введите **`(nth 1 endpt)`**↵. AutoLISP возвратит `2.0`.
8. Не сохраняйте чертеж.

Программирование условных выражений

Необходимость выполнить какую-либо процедуру при наличии некоторых условий возникает довольно часто. Условное выражение проще всего создать при помощи оператора `IF`, который выполняет одно действие, если выражение-операнд истинно, и другое, если оно ложно. Иными словами, результат операции зависит от истинности некоторого выражения.

В любом языке программирования одной из важнейших управляющих структур являются циклы, которые позволяют повторять определенную процедуру несколько раз, пока она не будет выполнена над всеми объектами или элементами. Оператор цикла устанавливает условия начала выполнения операций, количество объектов, над которыми выполняются операции, и условия выхода из цикла.

Условные выражения

Условные выражения позволяют управлять ходом выполнения программы на основе анализа некоторых данных. Результатом условного выражения будет либо `T` (истина), либо `nil` (ложь). Например, для оператора

```
(< 3 5)
```

AutoLISP возвратит T (истина), поскольку 3 меньше, чем 5. Для оператора

```
(> 3 5)
```

AutoLISP возвратит nil (ложь), поскольку 3 не больше 5. Для оператора

```
(= 3 5)
```

AutoLISP возвратит nil. Так как эти операторы возвращают T или nil, то их можно использовать в условном выражении оператора IF. В общем случае синтаксис оператора IF следующий:

```
(if условное_выражение Если_True Если_False).
```

Предположим, требуется выбрать окружности с радиусом меньше 0.25. Вот пример оператора IF, в котором radius — переменная с присвоенным ранее значением:

```
(if (< radius .25)
  (princ "Радиус меньше .25")
  (princ "Радиус не меньше .25")
)
```

Здесь используется условное выражение (< radius .25). Выражение *Если_True* — это (princ "Радиус меньше .25"), а выражением *Если_False* — (princ "Радиус не меньше .25"). Этот условный оператор эквивалентен следующему утверждению: *Если радиус меньше, чем .25, печатать "Радиус меньше .25", если нет — "Радиус не меньше .25"*.

Выражение *Если_False* можно опустить. Тогда AutoLISP выполнит выражение *Если_True*, если условное выражение истинно, а если оно ложно, сразу перейдет к выполнению остальной части программы. В следующем упражнении вы встретите оба типа операторов IF.

Пошаговая инструкция

Использование оператора IF

1. Создайте в AutoCAD новый чертеж, используя опцию Start from Scratch.
2. Откройте Visual LISP, создайте новый файл и введите следующий текст:

```
(defun c:compare2three ( / entered_num)
  (setq entered_num (getint "\nВведите число:"))
  (if (< entered_num 3)
     (princ "\nВведенное число меньше 3")
     (if (= entered_num 3)
       (princ "\nВведенное число равно 3")
       (princ "\nВведенное число больше 3")
     )
  )
(princ)
)
```

   Функция GETINT, получающая целое число от пользователя, будет описана в этой главе несколько ниже, как и оператор (princ). Благодаря наличию префикса \n в строке \nВведите число каждое новое приглашение будет выводиться с новой строки. Использование этого префикса аналогично применению оператора (terpri).

3. Щелкните на пиктограмме Check Edit Window (Проверить программу в окне редактирования). Если в окне Build Output (Результат) появятся сообщения об ошибках, исправьте код и повторите операцию.

4. Сохраните файл под именем `ab35-2.lsp` в папке, включенной в список поиска путей файлов поддержки, или в `Acad2000\Support\`.
5. Щелкните на пиктограмме Load active edit window, а затем на пиктограмме Activate AutoCAD.
6. Чтобы увидеть результат работы условного оператора `IF`, введите **compare2three**↵. В ответ на приглашение `Введите число:` введите **5**↵. AutoCAD ответит: `Введенное число больше 3`.
7. Повторите команду **COMPARE2THREE**. В ответ на приглашение `Введите число:` введите **3**↵. AutoCAD ответит: `Введенное число равно 3`.
8. Еще раз повторите команду **COMPARE2THREE**. В ответ на приглашение `Введите число:` введите **2**↵. AutoCAD ответит: `Введенное число меньше 3`.
9. Не сохраняйте чертеж.

Циклы

Циклы обеспечивают возможность выполнения одного или нескольких операторов программы заданное количество раз. Один из способов организации цикла — использование оператора `WHILE`.

Формат функции `WHILE` следующий:

```
(while условное_выражение Если_True
      операторы тела цикла, которые выполняются до тех пор, пока условие истинно)
```

Общий подход к установке условий состоит во включении *счетчика* в оператор `WHILE`. Счетчик подсчитывает количество выполненных операций. Условием выхода из цикла может служить достижение счетчиком некоторого значения. Чтобы сформировать счетчик, присвойте переменной значение, с которого вы хотите начать выполнение операции. Затем напишите код (точнее, его фрагмент) для одного прохода. После этого увеличьте значение счетчика, используя выражение типа

```
(setq counter (+ 1 counter))
```

Программа будет повторять выполнение операции до тех пор, пока счетчик (переменная `counter`) не достигнет установленного вами значения.

Вот простой пример:

```
(defun c:process (/ counter)
   (setq counter 1)
   (while (< counter 6)
      (princ "Номер цикла обработки ")
      (princ counter)
      (terpri)
      (setq counter (+ 1 counter))
   )
)
```

В этом примере функция `PROCESS` присваивает переменной `counter` значение `1`. Затем начинается цикл `WHILE`, условием выполнения которого является то, что значение переменной `counter` должно быть меньше `6`. Внутри оператора `WHILE` печатается строка `"Номер цикла обработки "` и затем значение переменной `counter`. Функция `(terpri)` обеспечивает печать каждой следующей строковой константы с новой строки. Затем для перемен-

ной counter вычисляется новое, большее значение. Каждый раз после выполнения операторов тела цикла значение счетчика увеличивается на 1. Без приращения значения переменной counter условие в строке 3 всегда будет возвращать T и цикл никогда не закончится.

Если вы случайно создали программу с таким бесконечным циклом, то можете прервать выполнение AutoLISP-программы, нажав <Esc>, <Ctrl+Break> или выбрав в меню Visual LISP команду Debug⇨Abort Evaluation.

В приведенном примере цикл WHILE продолжается до тех пор, пока переменная counter меньше 6. Когда значение счетчика станет равным 6, программа остановится. Оператор WHILE возвращает последнее значение программы, так что AutoCAD в последней строке выведет 6. На рис. 35.3 показан результат выполнения этой программы.

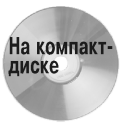
```
Command: process
Processing number 1
Processing number 2
Processing number 3
Processing number 4
Processing number 5
6
```

Рис. 35.3. Результат выполнения функции process

При работе с оператором WHILE может возникнуть желание объединить несколько операций в один блок. Обычно функция IF при условии истинности логического выражения выполняет один оператор. Однако вам может понадобиться при выполнении этого условия осуществить несколько операций. Если условный оператор состоит из двух частей — *Если_True* и *Если_False*, то нужно каким-то образом отделить эти группы операторов друг от друга. Для этого можно использовать функцию PROGN. AutoLISP, встретив эту команду, будет рассматривать все, что включено в конструкцию PROGN, как один оператор.

В следующем примере вы увидите использованные ранее условные операторы IF. Однако во втором операторе, если введено число 3, будет выведена не одна, а две строки. Это достигается объединением соответствующих строк кода оператором PROGN.

```
(defun c:compare2three (/ entered_num)
  (setq entered_num (getint "\nВведите число: "))
  (if (< entered_num 3)
    (princ "\nВведенное число меньше 3.")
    (if (= entered_num 3)
      (progn
        (princ "\n Введенное число равно 3.")
        (terpri)
        (princ "\nЭто и есть результат эксперимента.")
      )
      (princ "\n Введенное число больше 3.")
    )
  )
  (princ)
)
```



Файл ab35-b.lsp, используемый в упражнении по применению операторов WHILE, IF, PROGN и счетчика, содержится на прилагаемом компакт-диске. Если вы еще не поместили этот компакт-диск в дисковод CD-ROM, сделайте это сейчас.

Пошаговая инструкция

Использование операторов WHILE, IF, PROGN и счетчика counter

1. Откройте AutoCAD и создайте новый чертеж.
2. Откройте Visual LISP.
3. Щелкните на пиктограмме Open File (Открытие файла) панели инструментов Standard и откройте файл ab35-b.lsp, расположенный на прилагаемом компакт-диске.

4. Сохраните файл под именем `ab35-3.lsp` в папке, включенной в список поиска путей файлов поддержки, или в `\AutoCAD R15\Support`. Текст этой программы показан на рис. 35.4.

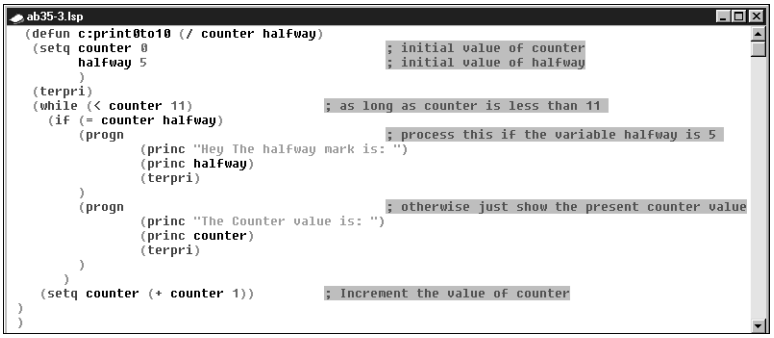


Рис. 35.4. Программа `print0to10`

5. Загрузите `ab35-3.lsp`. Вернитесь в AutoCAD.
6. Введите **`print0to10`**↵. Нажмите клавишу <F2> — откроется текстовое окно AutoCAD и будет виден результат, показанный на рис. 35.5.

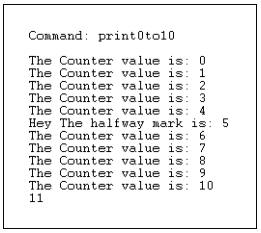


Рис. 35.5. Результат работы функции `print0to10`

7. Не сохраняйте чертеж.

Работа с объектами чертежей

Истинная мощь языка AutoLISP проявляется при работе с объектами чертежей. В этом разделе описаны AutoLISP-программы, выполняющие эти магические действия.

Получение информации об объектах

Любой объект базы данных AutoCAD (или примитив) имеет имя. По этому имени можно получить справку об объекте из любого места собственного AutoLISP-приложения. Чтобы увидеть пример имени объекта, после создания нового чертежа введите

```
(command "_line" "3,3" "5,5" "")↵
(entlast)↵
```

AutoLISP возвратит <Entity name:2ed0520> (<Имя примитива:2ed0520>).

Возможно, в вашей системе имя примитива будет определяться другим числом. Используя возвращаемое функцией ENTLAST имя примитива, можно программным путем получать или устанавливать набор опций для любого заданного объекта базы данных.

Функция ENTGET (*ENTity GET*) (Получить примитив) — это ключ к изменению базы данных чертежа. В качестве единственного аргумента эта функция получает имя примитива. После вычерчивания отрезка в приведенном выше примере введите

```
(setq myline (entget (entlast)))↵
```

AutoLISP возвратит:

```
((-1 . <Entity name: 15ac558>) (0 . "LINE") (330 . <Entity name:
15ac4f8>)
(5 . "2B") (100 . "AcDbEntity") (67 . 0) (410 . "Model") (8 . "0")
(100 . "AcDbLine") (10 3.0 3.0 0.0) (11 5.0 5.0 0.0) (210 0.0 0.0
1.0))
```

В таком виде отрезок хранится в базе данных чертежа. AutoLISP возвратил один большой список, содержащий 10 вложенных. К каждому из вложенных списков (подсписков) можно обращаться как группе, индексированной по ее первому элементу. Каждый из подсписков состоит из двух частей. Первая часть — код группы данных, входящих во вторую часть. Имя примитива входит в группу с кодом -1. Каждый код группы отвечает за определенное свойство отрезка. Перечень наиболее часто используемых кодов групп приведен в табл. 35.6.

Таблица 35.6. Коды групп стандартных объектов AutoCAD

Код группы	*Описание*
-1	Имя примитива
0	Тип примитива
1	Текстовая константа
8	Слой
10	Начальная точка (или центр)
11	Конечная точка (или точка выравнивания)
38	Высота
39	Толщина
40	Радиус (или высота текста)
62	Цвет
67	Флаг пространства листа

Для изучения объекта AutoCAD можно также использовать Visual LISP. Для этого сначала выберите команду View⇨Browse drawing database⇨Browse Selection (Вид⇨Просмотр базы данных чертежа⇨Просмотр выделенного фрагмента). Затем выделите нужный объект.

После этого в Visual LISP откроется диалоговое окно Inspect (Инспектор). Выделите имя примитива и, удерживая указатель мыши над выделенным текстом, щелкните правой кнопкой. Выберите в контекстном меню команду Inspect, чтобы просмотреть информацию об объекте. На рис. 35.6 представлена информация о созданном ранее отрезке.

В списке вычерченного отрезка присутствуют не все коды групп. Например, в списке, который возвратил AutoLISP, отсутствует группа 62 (цвет). При вычерчивании отрезка вы не задали явно его цвет. В результате для него по умолчанию был установлен текущий цвет. По этой же причине AutoLISP устанавливает не все атрибуты для каждой группы явно. В данном

случае цвет установлен функцией ByLayer (ПОСЛОЮ), а текущим слоем является слой 0. AutoLISP возвратил (8 . "0") в приведенном выше списке, чтобы подчеркнуть, что отрезок находится в слое 0.

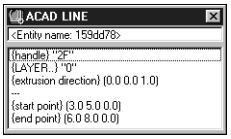


Рис. 35.6. Получение информации об объекте чертежа в Visual LISP

Кроме приведенных в табл. 35.6, имеется много других кодов групп. Эти коды можно найти в справочной системе Visual LISP, выбрав в Visual LISP команду Help⇨Visual LISP Help Topics (Справка⇨Разделы справки по Visual LISP). Перейдите на вкладку Contents (Оглавление) и дважды щелкните на элементе DXF Reference (Руководство по DXF). (Поскольку коды групп используются также в DXF-файлах, они описаны в руководстве по DXF). Щелкните дважды на элементе Chapter 6 - ENTITIES Section. Затем можно выбрать либо раздел Common Group Codes for Entities (Стандартные коды групп для примитивов), либо конкретный интересующий вас примитив. Во многих случаях один и тот же код группы может иметь различное значение в зависимости от примитива, к которому он относится. Например, в приведенном выше списке для начерченного отрезка группа 10 представлена данными (3.0 3.0 0.0), которые означают, что начальная точка отрезка имеет координаты X=3.0, Y=3.0, Z=0.0. Если бы группа с кодом 0 описывала окружность, то координаты группы 10 определяли бы центр этой окружности.

Выбор и изменение различных данных, относящихся к примитиву, осуществляется функциями ASSOC и SUBST.

- Функция ASSOC возвращает подсписок, найденный по входу, ассоциированному с элементом списка. Аргументами функции ASSOC являются элемент и сам список. Например, если в качестве первого аргумента определить код группы (скажем, 10), она возвратит значение кода (в данном случае — начальную точку отрезка). Если список с именем myobject содержит три группы — ((0 . "group 0") (1 1.0 2.0) (3 4.2 1.5 6.75)),,то,(assoc 1 myobject),возвратит,(1 1.0 2.0).
- Функция SUBST заменяет в списке одно значение другим. Она использует три аргумента. Первый определяет, *чем заменять*, второй — *что заменять*, третий — *в каком подсписке* выполнить замену.

Рассмотрим работу этих функций на примере изменения координат начальной точки отрезка.

```
(setq startpt (assoc 10 myline))
```

AutoLISP возвратит: (10 3.0 3.0 0.0).

Изменим начальную точку:

```
(setq new_startpt '(10 6.5 1.0 0.0))
(setq myline (subst new_startpt startpt myline))
```

AutoLISP возвратит:

```
((-1 . <Entity name: 15ac558>)
 (0 . "LINE") (330 . <Entity name: 15ac4f8>)
 (5 . "2B") (100 . "AcDbEntity")
 (67 . 0) (410 . "Model")
 (8 . "0") (100 . "AcDbLine")
 (10 6.5 1.0 0.0) (11 5.0 5.0 0.0) (210 0.0 0.0 1.0))
```

В данном случае существующая переменная startpt объекта myline заменена на new_startpt. Непосредственно в отрезок никакие изменения пока что не внесены. Для внесения изменений в объект используется функция ENTMOD.

Изменение объектов

Ключом для изменения объектов служит функция ENTMOD (*ENTity MODify*) (Редактировать примитив). Список, возвращенный AutoLISP, можно изменить, а затем передать функцию ENTMOD в качестве аргумента для обновления базы данных AutoCAD. Продолжим рассмотренный выше пример. Введите

```
(entmod myline)
```

AutoLISP возвратит:

```
((-1 . <Entity name: 15ac558>)
 (0 . "LINE")(330 . <Entity name: 15ac4f8>)
 (5 . "2B") (100 . "AcDbEntity")
 (67 . 0) (410 . "Model")
 (8 . "0") (100 . "AcDbLine")
(10 6.5 1.0 0.0) (11 5.0 5.0 0.0) (210 0.0 0.0 1.0))
```

В базу данных AutoCAD внесены изменения, и теперь начальной точкой отрезка является точка с координатами X=6.5, Y=1.0, Z=0.0.

Создание набора выбранных объектов

С помощью функции SSGET (*Selection Set GET*) (Получение набора) можно создать набор объектов. Эта функция выдает уже знакомое вам приглашение Select objects:. Наиболее часто с наборами объектов используются функции, приведенные в табл. 35.7.

Одновременно можно использовать не более 256 наборов. Чтобы освободить набор в AutoLISP для повторного использования в дальнейшем, нужно установить его в nil, например (setq ss nil).

Таблица 35.7. Стандартные функции работы с наборами объектов AutoCAD

Функция	*Описание*
SSGET	Обрабатывает набор, полученный от пользователя
SSLENGH	Возвращает число объектов в наборе. Для этой функции требуется один аргумент — набор
SSNAME	Возвращает имя примитива данного объекта в наборе. Для ее работы требуются два аргумента: набор и номер объекта в наборе

В качестве примера для нового чертежа можно ввести следующую программу:

```
(command "_circle" "3.3" "2")
nil
(command "_circle" "4.4" "3")
nil
(command "_line" "7.2" "6.6" "3.4" "5.5" "")
nil
(setq mysset (ssget))
Select objects: all↵
5 found
Select objects:↵
<selection set 1>
```

В рассмотренном примере переменной mysset присвоен набор, сформированный опцией **all**, который содержит три отрезка и два круга. Чтобы увидеть содержимое набора, введем в командной строке или в окне Console Visual LISP следующий код:

```
(sslength mysset)↵
5
```

Итак, теперь вы убедились, что набор содержит пять объектов. Первому объекту присваивается номер 0, а пятому — 4. Посмотрим, что собой представляет первый объект:

```
(ssname mysset 0)↵
<Entity name: 3fe0550>
```

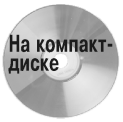
Для получения данных об объекте введем

```
(entget (ssname mysset 0))↵
```

AutoLISP возвращает:

```
((-1 . <Entity name: 1601580>)
 (0 . "LINE") (330 . <Entity name: 16014f8>)
 (5 . "30") (100 . "AcDbEntity")
 (67 . 0) (410 . "Model")
 (8 . "0") (100 . "AcDbLine")
 (10 3.0 4.0 0.0) (11 5.0 5.0 0.0) (210 0.0 0.0 1.0))
```

Такие же действия можно выполнить и с другими именами примитивов, входящих в набор.



В файле \Software\Chap35\Quicklsp на прилагаемом компакт-диске содержится справочная таблица с описанием многих полезных функций AutoLISP.

Пошаговая инструкция

Создание набора объектов чертежа

1. Создайте новый чертеж в режиме Start from Scratch (Без шаблона) и введите в новый файл, открытый в окне редактирования Visual LISP, следующий ниже код. Сохраните файл под именем ab35-4.lsp в папке, включенной в список доступа к файлам поддержки, или в \Acad2000\Support.

```
(defun c:listsset ( / mysset counter)
  (setq mysset (ssget))
  (setq counter 0)
  (while (< counter (sslength mysset))
         (terpri)
          (princ (cdr (assoc 0 (entget (ssname mysset counter)))))
             (setq counter (+ counter 1))
          )
       (princ)
  )
```

2. Загрузите ab35-4.lsp.
3. Активизируйте AutoCAD и начертите несколько объектов на экране (не менее двух различных типов).
4. Введите **listset**↵. При выполнении функции SSGET AutoCAD предложит вам выбрать объекты.

5. Выберите все объекты чертежа. Программа выведет типы каждого из них. (Нажмите клавишу <F2>, чтобы открыть окно AutoCAD Text Window и увидеть результат, показанный на рис. 35.7.) Конечно, ваш результат может отличаться от показанного на рисунке, так как, возможно, были вычерчены объекты других типов.

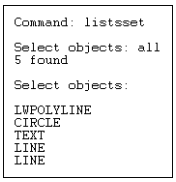


Рис. 35.7. Возможный результат выполнения функции `listsset`

Рассмотрим, как работает эта программа.

- В первой строке создается функция и объявляются две переменные: `mysset` и `counter`.
- Во второй строке переменной `mysset` присваивается набор, созданный пользователем по запросу функции `SSGET`.
- В третьей строке счетчик `counter` устанавливается в `0`.
- В четвертой строке начинается цикл `WHILE`. Выполнение операций начинается с самой внутренней пары скобок. Сначала функция `SSLENGTH` определяет количество объектов в наборе `mysset`. Затем задается условие выполнения операторов тела цикла: цикл будет выполняться до тех пор, пока значение счетчика не превышает число объектов в наборе `mysset`. Другими словами, цикл прекращается после обработки всех объектов в наборе.
- В пятой строке функция `TERPRI` осуществляет переход на новую строку, в которой будет выведен список объектов.
- В шестой строке кода выводится на экран содержимое списка. Как обычно, выполнение операций начинается с самой внутренней пары скобок. Поэтому сначала определяется имя объекта в наборе `mysset`, номер которого соответствует текущему значению счетчика. Затем с помощью функции `ENTGET` программа получает объект. После этого функция `ASSOC` извлекает имя объекта из группы с кодом `0`. В результате получается точечная пара, вторым элементом которой является имя объекта. Функция `CDR` пропускает первый элемент точечной пары, оставляя лишь нужное нам имя объекта. И наконец, программа выводит результат.
- В седьмой строке значение счетчика увеличивается на `1` и цикл `WHILE` продолжается для следующего объекта.
- В восьмой строке закрывается цикл `WHILE`.
- В девятой строке программа завершается. Вопросы корректного завершения обсуждаются несколько ниже в этой главе.
- В десятой строке стоит скобка, ограничивающая тело функции.

Ввод данных пользователем

Часто в процессе выполнения AutoLISP-программ включается ввод данных пользователем. Чтобы обеспечить эту связь, в AutoLISP содержится ряд функций с префиксом `GET`. Например, функция `GETVAR` обрабатывает информацию о системных переменных. В табл. 35.8 перечислены некоторые другие полезные `GET`-функции.

Таблица 35.8. Основные функции для ввода данных пользователем

Функция	*Описание*
GETDIST	Возвращает расстояние между двумя точками, указанными пользователем
GETINT	Возвращает целое число
GETREAL	Возвращает действительное число
GETSTRING	Возвращает текстовую константу

Для приостановки выполнения программы и ввода значений или указания точки на объекте в тело функции COMMAND можно включить функцию PAUSE. Например, выражение (command "circle" pause "3") делает паузу для того, чтобы пользователь определил центр, а затем создает окружность радиусом 3.

Обратите внимание на новую функцию ENTSEL в следующем упражнении. Она представляет собой упрощенный вариант функции SSGET и используется для выбора пользователем одного объекта. Функция ENTSEL возвращает имя примитива и точечную пару, содержащую координаты указанной точки. Следовательно, после использования функции ENTSEL с помощью функции CAR можно получить имя примитива для функции ENTGET.

В упражнении вы также встретитесь с новым аргументом T функции getstring. Если он используется и не равен nil, то при вводе данных допускаются пробелы. Таким образом, этот аргумент позволяет пользователю вводить текстовые константы, состоящие из нескольких слов.

Пошаговая инструкция

Получение данных от пользователя

1. Создайте новый чертеж в режиме Start from Scratch.
2. Откройте Visual LISP, создайте новый файл и введите следующий код. Сохраните файл под именем ab35-5.lsp в папке Acad2000\Support или любой другой папке, включенной в список поиска путей файлов поддержки.

```
(defun c:chgmytext ( / src_object new_ht new_str)
  (setq src_object (entget (car (entsel))))
  (setq new_ht (getreal "\n Какова новая высота текста? "))
  (setq new_str (getstring T "Новый текст: "))
  (setq scr_object
    (subst (cons 40 new_ht) (assoc 40 src_object) src_object)
  )
  (setq scr_object
    (subst (cons 1 new_str) (assoc 1 src_object) src_object)
  )
  (entmod src_object)
  (princ)
)
```

3. Загрузите файл ab35-4.lsp.
4. С помощью команды **DTEXT** (ДТЕКСТ) или **TEXT** (ТЕКСТ) создайте какой-нибудь фрагмент текста и выполните над ним функцию chgmytext. В ответ на приглашения введите значения высоты и новый текст. AutoCAD изменит высоту и содержимое текстового объекта.
5. Не сохраняйте чертеж.

Эта программа работает следующим образом.

- В первой строке определяется функция и объявляются три переменных.
- Во второй строке функция ENTSEL предоставляет пользователю возможность выбрать один объект. Как уже упоминалось, функция CAR выделяет имя примитива, которое затем передается в функцию ENTGET для модификации. После этого окончательное имя примитива присваивается переменной src_object.
- В третьей строке выводится приглашение задать новую высоту текста и введенное пользователем число присваивается переменной new_ht.
- В четвертой строке выводится приглашение задать новый текст и введенное пользователем значение присваивается переменной new_str.
- В пятой строке начинается процесс замены старых значений новыми. Здесь запускается функция, присваивающая новые значения параметров объекту src_object.
- В шестой строке функция SUBST заменяет для объекта src_object старое значение высоты текста новым. (Код группы 40 соответствует высоте текста.)
- В седьмой строке завершается вызов функции SETQ.
- Восьмая строка аналогична пятой. Операции, выполненные для высоты текста, теперь повторяются для его содержания.
- В девятой строке функция SUBST заменяет старое содержание текста объекта src_object новым. (Код группы 1 соответствует значению текстовой константы.)
- В десятой строке завершается вызов функции SETQ.
- В одиннадцатой строке функция ENTMOD вносит изменения в базу данных чертежа.
- В двенадцатой строке программа завершается.
- В тринадцатой строке стоит скобка, ограничивающая тело функции.

Последние штрихи

Нашу программу нельзя считать завершенной, если не добавить к ней еще несколько заключительных штрихов.

Вы, наверное, обратили внимание, что в конце нескольких программ использовалась функция PRINC. Это означает, что при завершении работы программы в командной строке не будет выводиться никаких дополнительных сообщений. Такой способ завершения программ называется *спокойным* или *чистым* (exiting cleanly или quietly).

В рассмотренных до сих пор приложениях очень мало внимания уделялось вопросам обработки ошибок. В четвертой строке программы, приведенной ниже, в пошаговой инструкции, используется новая функция проверки на равенство — EQUAL. Эта функция отличается от оператора = тем, что она возвращает T только в том случае, когда два выражения равны (каждый из объектов перед проверкой на равенство предварительно вычисляется). Чтобы не спутать эти конструкции, проще всего запомнить, что для сравнения списков применяется функция EQUAL, а для сравнения чисел — оператор =.

В программе, приведенной ниже, в пошаговой инструкции, предусмотрен вариант обработки ошибочного выбора нетекстового объекта. Если выбран объект, не являющийся текстовым, программа перейдет к выполнению строки 18 и напечатает сообщение об ошибке: Вы должны выбрать текстовый объект.

Аналогичную обработку ошибок при выборе объектов можно проводить с использованием функции IF. Для выполнения этой функции требуется наличие логического выражения. Поэтому, если пользователь не выберет объект нужного типа, логическое выражение должно принять значение false. Во вторую часть ЕСЛИ_FALSE условного оператора IF можно поместить сообщение об ошибке.

Подобная обработка ошибок необходима для правильного функционирования AutoLISP-программ.

Еще один заключительный штрих — включение в текст программы комментариев. В начале программы включите комментарий с описанием ее функционального назначения. Это поможет другим пользователям вашей программы лучше понять ее логику, а вам самим — быстрее вспомнить собственный замысел несколько месяцев спустя. Строки комментария в AutoLISP-программах начинаются с символа "точка с запятой" (;).

Комментарии также следует помещать и в тексте программы. Отсутствие комментариев может испортить даже самый лучший код. Большинство профессиональных программистов очень подробно комментируют и документируют свои программы.

Visual LISP поддерживает несколько типов комментариев. Щелкните на пиктограмме Format Edit Window панели инструментов Tools, и Visual LISP автоматически выровняет комментарии этих типов в вашей программе в процессе форматирования кода. Рассмотрим их подробнее.

- **;;; — три точки с запятой**. При использовании такого типа комментария Visual LISP выравнивает комментарий по левому краю. Этот тип комментария используют в начале программы для описания ее назначения и функционирования.
- **;; — две точки с запятой**. Этот комментарий сдвигается в процессе форматирования на уровень следующих скобок. Такой комментарий используется для пояснения следующей за ним одной или нескольких строк кода.
- **; — точка с запятой**. Комментарий, помеченный этим символом, Visual LISP в процессе форматирования кода по умолчанию сдвигает на 40 символов. Это значение можно изменить в диалоговом окне Format Options (Параметры форматирования), которое открывается при выборе команды Tools⇨Environment Options⇨Visual LISP Format Options (Сервис⇨Параметры среды⇨Параметры форматирования Visual LISP) (рис. 35.8). Этот тип комментария обычно используется для текущих пояснений в правой части кода. Благодаря большому отступу такие комментарии хорошо выделяются на фоне кода.
- **;| |; — внутренний комментарий**. Внутренние комментарии можно поместить внутри любой строки кода. Внутренний комментарий форматируется следующим образом: `;| Это комментарий |;`. Текст комментария ограничивается точкой с запятой и вертикальной чертой. Внутренний комментарий используется для пояснения небольшого фрагмента кода, расположенного в одной строке.

Ниже приведены примеры комментариев различного типа.

Использование AutoLISP для копирования свойств объектов

В следующей программе выбранному объекту присваивается тот же слой, что и другому объекту. Многие пользователи AutoCAD активно применяли эту программу до появления функции MATCHPROP в версии AutoCAD R14. Общая методология, выбранная здесь для копирования слоев объектов, может быть использована для изменения любых других свойств объектов AutoCAD.

```
;;;Копирование слоя объекта и присвоение его другим
;;;  выбранным объектам
(defun c:matchlayer (/ src_object mysset counter
cur_ent_ent_layer)
  ;; приглашение пользователю
  (princ "\n*** Выберите исходный объект ***")
  (if (setq src_object (car (entsel)))     ; выбрать объект
    (progn
                                        ; считывание слоя
                                        ; выбранного объекта
      (setq src layer (assoc 8 (entget src object)))
      ;; приглашение пользователю
      (princ "\n*** Выберите объекты для копирования
слоя***")
      ;; выбор нескольких объектов с использованием
      ;; функции ssget
      (if (setq mysset (ssget)) ; анализ выбранных объектов
        (progn     ; если объекты выбраны, выполнить следующее
          (setq counter 0)
          (while (< counter (sslength mysset))

            (setq cur_ent (entget (ssname mysset counter)))
            (setq ent layer (assoc 8 cur ent))
            (entmod (subst src_layer ent_layer cur_ent))
            (setq counter (+ counter 1))
          )    ; конец while
       )  ; конец progn
       (princ "\nНи одного объекта не выбрано")  ; сообщение
об ошибке
      )      ; конец if
    )       ; завершение функции progn для выбора объектов
    ;; сообщение пользователю, если не выбран исходный объект
    (princ "\nВы не выбрали исходный объект")
  )         ; конец оператора if
  (princ)
)           ; конец функции c:matchlayer
```

В этой программе сначала определяется имя выбранного объекта с использованием функции `(car (entsel))`. Затем функция `ENTGET` выбирает сам объект по его имени, а функция `ASSOC` определяет его слой в группе с кодом `8`. Затем в программе создается набор из выбранных объектов. В цикле для каждого из этих объектов определяется его слой (переменная `ent_layer`) и с помощью функций `ENTMOD` и `SUBST` текущее значение слоя заменяется значением слоя исходного объекта.

Обработка каждой пары объектов (объекта-донора, слой которого копируется, и объектов-реципиентов) осуществляется в теле функции `IF`, оператор `ЕСЛИ-FALSE` которой выдает сообщение об ошибке, если не выбран хотя бы один из необходимых объектов.

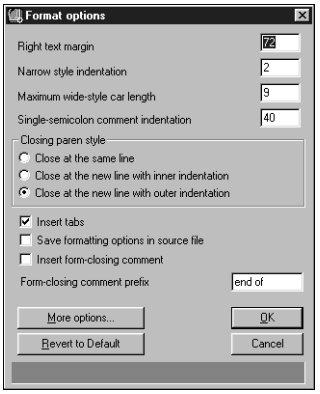


Рис. 35.8. В диалоговом окне Format Options редактора Visual LISP можно изменить отступы и другие параметры форматирования кода

Пошаговая инструкция

Завершающие штрихи

1. Загрузите программу, созданную в предыдущем упражнении. Если вы его не делали, введите программу в окне редактора Visual LISP и сохраните файл под именем `ab35-5.lsp` в папке `\AutoCAD R15\Support` или любой другой папке, включенной в список поиска путей файлов поддержки. Затем откройте в AutoCAD любой чертеж и загрузите текст программы.
2. Выполните функцию `chgmytext` и в ответ на приглашение `Select object:` (Выберите объект:) выберите объект, не являющийся текстовым (например, `circle`). В ответ на приглашения введите высоту и новый текст.

   Если это проделать по отношению к кругу, то обозначение его радиуса изменится в соответствии с указанной новой высотой текста. Это, конечно, совсем не то, что имелось в виду при написании этой программы.
3. Измените программу, как указано ниже, и сохраните ее в файле `ab35-6.lsp`:

```
;;;изменение высоты текста и его содержания
(defun c:chgmytext ( / src_object new_ht new_str)
  (terpri)
  (setq src_object (entget (car (entset))))
  (if (equal (assoc 0 src_object) '(0 . "TEXT"))
    (progn
      (princ "Какова новая высота текста? ")
      (setq new_ht (getreal))
      (princ "Новый текст: ")
      (setq new_str (getstring))
      (setq scr_object
         (subst (cons 40 new_ht) (assoc 40 src_object)
               src_object)
      )
      (setq scr_object
          (subst (cons 1 new_str) (assoc 1
```

```
src_object)src_object)
      )
      (entmod src_object)
    )  ; конец progn
    (princ "Вы должны выбрать текстовый объект.")
  ) ; конец if
  (princ)
)   ; конец c:chgmytext
```

4. Загрузите программу `ab35-6.lsp` и снова выполните программу с кругом или другим нетекстовым объектом.
5. Не сохраняйте чертеж.

Резюме

Прочитав эту главу, вы научились создавать переменные, функции AutoLISP, а также работать с командами AutoCAD и системными переменными. Вы расширили свои знания об AutoLISP, познакомившись с использованием списков и циклов. Из этой главы вы узнали, как получить и изменить информацию об объектах чертежа, научились создавать наборы объектов. Используя эти методы, а также ввод данных пользователем, можно автоматизировать процесс модификации объектов чертежа.

Заканчивая работу над AutoLISP-программой, всегда следует добавлять обработку ошибок, чтобы обеспечить спокойное завершение программы, а также комментарии.

Из следующей главы вы узнаете о расширенных возможностях Visual LISP.