



Министерство образования Республики Беларусь

Учреждение образования
«Гомельский государственный технический
университет имени П. О. Сухого»

Кафедра «Технология машиностроения»

ИСПОЛЬЗОВАНИЕ ЯЗЫКА AUTOLISP ДЛЯ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

**Лабораторный практикум
по курсу «Основы автоматизированного
проектирования» для студентов специальностей
1-36 01 01 «Технология машиностроения»
и 1-36 01 03 «Технологическое оборудование
машиностроительного производства»
дневной и заочной форм обучения**

Электронный аналог печатного издания

Гомель 2007

УДК 004.43(075.8)
ББК 32.973-018.1я73
И88

*Рекомендовано к изданию научно-методическим советом
машиностроительного факультета ГГТУ им. П. О. Сухого
(протокол № 3 от 28.11.2005 г.)*

Автор-составитель: *В. С. Мурашко*

Рецензент: начальник сектора разработки средств АСУ ГГТУ им. П. О. Сухого
Н. С. Шестакова

И88 **Использование** языка AUTOLISP для автоматизированного проектирования :
лаб. практикум по курсу «Основы автоматизированного проектирования» для студентов
специальностей 1-36 01 01 «Технология машиностроения» и 1-36 01 03 «Технологическое
оборудование машиностроительного производства» днев. и заоч. форм обучения
/ авт.-сост. В. С. Мурашко. – Гомель : ГГТУ им. П. О. Сухого, 2007. – 35 с. – Систем. тре-
бования: PC не ниже Intel Celeron 300 МГц ; 32 Mb RAM ; свободное место на HDD
16 Mb ; Windows 98 и выше ; Adobe Acrobat Reader. – Режим доступа: <http://gstu.local/lib>. –
Загл. с титул. экрана.

ISBN 978-985-420-563-2.

Лабораторный практикум способствует усвоению основ программирования на языке Autolisp,
иллюстрированных примерами, для выполнения лабораторных работ. Рассмотрены алгоритмы рабо-
ты программ-параметризаторов.

Для студентов, изучающих курс «Основы САПР» дневной и заочной форм обучения.

УДК 004.43(075.8)
ББК 32.973-018.1я73

ISBN 978-985-420-563-2

© Мурашко В. С., составление, 2007
© Учреждение образования «Гомельский
государственный технический университет
имени П. О. Сухого», 2007

Введение

Настоящее практическое пособие используется для выполнения лабораторных работ по курсу «Основы САПР» для студентов специальностей 1-36 01 01 «Технология машиностроения» и 1-36 01 03 «Технологическое оборудование машиностроительного производства».

Цель данного пособия: дать основы программирования на языке AutoLISP и параметрического проектирования.

Главное предназначение системы AutoCAD – не рисование чертежей на компьютере, а создание на ее основе специализированной САПР определенного класса изделий.

Анализ работы конструкторско-технологических служб ряда промышленных предприятий позволил установить, что одна из наиболее трудоемких проектных процедур в ходе КТПП – разработка конструкторской документации на ряд близких по конструкции деталей и/или сборочных единиц (ДСЕ), отличающихся в основном своими размерными параметрами или вариантами исполнения. Данная процедура является трудоемким и нетворческим процессом с низкой производительностью и высокой вероятностью внесения ошибок. Особенно часто требуется выпускать конструкторскую документацию на средства технологического оснащения (СТО) машиностроительного производства: тиски, кондукторы, пресс-формы и т. д., причем подготовка этой документации должна вестись опережающими темпами для обеспечения времени на изготовление СТО к моменту запуска изделия в производство.

Данную проблему можно решить с помощью параметрического проектирования, сущность которого состоит в создании математической модели класса конструктивно однородных изделий, а затем в генерации изображений этих изделий по набору задаваемых размерных параметров.

Студентам предлагаются лабораторные работы по созданию программ-параметризаторов на языке AutoLISP.

1. Классификация языков программирования.

Язык AutoLISP

Классификация языков, представленная на рис. 1.1, несколько условна, поскольку почти все они содержат те или иные элементы друг друга и со временем взаимно дополняются и обогащаются.



Рис. 1.1. Классификация языков программирования

Язык LISP создан американским ученым Джоном Маккарти и нашел широкое применение. Он отличается высокой надежностью, функциональным стилем программирования и использованием обратной польской нотации. Основное понятие языка LISP – список.

Список – перечень атомов или списков, отделенных друг от друга пробелами и заключенных в скобки. Списки могут быть вложенными.

Атом – простой (в отличие от списка) тип данных: число, символьная строка, функция.

В LISP нет различия между текстом программы и обрабатываемыми ею данными. В LISP и данные, и текст программы являются списками.

Используется специальная система записи списка – обратная польская нотация. Вызов любой функции в данной нотации записывается как список следующего вида:

$(\text{имя_функции } a_1 a_2 \dots a_n),$

где $a_1 a_2 \dots a_n$ – аргументы функции.

Например, если функция сложения двух чисел имеет имя "+", то операция $2 + 3$ запишется как $(+ 2 3)$. В качестве аргументов могут фигурировать другие функции, что позволяет записывать сколь угодно сложные формулы в обратной польской нотации.

Пусть дана функция $f(x, y) = \left(\frac{2x+3}{y-1} \right) \cdot (y-x)$. В обратной польской записи она приобретет следующий вид:

Полученная запись имеет вид:

$(* (/ (+ (* 2 x) 3) (- y 1)) (- y x))$.

Рассмотрим конкретную реализацию языка LISP – встроенный в САПР AutoCAD интерпретатор языка AutoLISP. Выбор этого языка в качестве встроенного для САПР AutoCAD вызван тем, что *список* – оптимальный способ представления графической информации, а также легкостью реализации и небольшими размерами интерпретатора.

Атом в AutoLISP представляет собой ссылку (адрес) ячейки памяти, начиная с которой записана та или иная информация.

Каждый атом имеет *имя*, дающееся по следующим правилам: допускаются английские буквы, цифры, большинство имеющихся на клавиатуре знаков за исключением ";", "(", ")", ".", ",", " ", строчные и заглавные буквы не различаются, первым символом должна быть буква.

Буква *T* зарезервирована и *не должна использоваться* в качестве имени атома. Слово *NIL* зарезервировано и *не должно использоваться* в качестве имени атома.

Длина имени формально не ограничена, но для экономии памяти рекомендуется не превышать длину в шесть знаков.

AutoLISP поддерживает следующие типы данных:

- целое число со знаком от -32768 до 32767 или от 0 до 65535 (2 байта) без знака;
- вещественное число, записываемое через десятичную точку: 10.52 или в экспоненциальном формате: 2.52E-12;
- строка символов длиной до 127 знаков, заключенная в двойные кавычки. Символ "\" является служебным и, если он нужен в тексте, должен удваиваться: текст "3\2" запишется как "3\\2";
- логический тип, принимающий два возможных значения: истина (обозначается *T*) или ложь (обозначается *NIL*);

- ссылка на встроенную функцию языка;
- ссылка на созданный программистом список (программу или данные);
- ссылка на переменную;
- ссылка на таблицу диспетчера виртуальной памяти.

2. Присваивание значений в AutoLISP. Встроенные функции

Следует различать два понятия: *переменная* и *значение переменной*. *Переменная* – указатель на область динамической памяти, имеющий имя.

Значение переменной – данные, записанные в динамической памяти, начиная с адреса, записанного в переменной.

Для присваивания значений переменным в AutoLISP имеются две функции – *SET* и *SETQ*.

Функция *SETQ* меняет значение переменной, а не саму переменную. Аргументами функции является перечень пар «переменная» – «значение». Функция возвращает результат последнего присваивания.

Например, запись (*SETQ a 10*) помещает число 10 в область памяти, на которую указывает переменная *a* и одновременно задает тип переменной *a* – целое число. Можно в одной функции присвоить несколько переменных. Порядок выполнения нескольких присваиваний в функции *SETQ* определен слева направо.

Изменение значения переменной на *NIL* освобождает занимаемую ее значением область памяти.

Функция *SET* – работает не со значениями, а с самими переменными. Например, (*SET a b*) заставляет переменную *a* ссылаться на ту же область памяти, что и переменная *b*.

Функция *QUOTE* запрещает вычисление списка, являющегося ее аргументом и возвращает сам этот список.

Например, (*SETQ a (QUOTE (0.25 "АБВ" 46))*), тогда значением переменной *a* станет список (0.25 "АБВ" 46). Введена сокращенная запись функции *QUOTE* в виде апострофа: (*SETQ a '(0.25 "АБВ" 46)*).

Функция *EVAL* вычисляет список, являющийся ее аргументом, и возвращает вычисленное значение.

Рассмотрим математические и ряд других встроенных функций AutoLISP, которые для удобства сведены в таблицу 2.1.

Пример. Вычислить площадь треугольника S со сторонами a, b, c по формуле Герона:

$$S = \sqrt{p(p-a)(p-b)(p-c)}, \quad p = \frac{a+b+c}{2}.$$

Программа вычислений:

```
(SETQ p (/ (+ a b c) 2))
(SETQ s (EXPT (* p (- p a) (- p b)
(- p c)) 0.5))
```

В языке также предусмотрена встроенная переменная PI , со значением, равным числу π .

Таблица 2.1

Основные встроенные функции AutoLISP

Имя функции	Аргументы	Возвращаемое значение
+	$a_1 a_2 \dots a_n$	$a_1 + a_2 + \dots + a_n$
-	$a_1 a_2 \dots a_n$	$a_1 - a_2 - \dots - a_n$
*	$a_1 a_2 \dots a_n$	$a_1 a_2 \dots a_n$
/	$a_1 a_2 \dots a_n$	$a_1 / a_2 / \dots / a_n$
1+	a	$a + 1$
1-	a	$a - 1$
ABS	a	$ a $
SQRT	a	\sqrt{a}
EXP	a	e^a
EXPT	$a b$	a^b
GCD	$a b$	НОД чисел a, b
LOG	a	$\ln(a)$
MIN	$a_1 a_2 \dots a_n$	Минимальное из чисел $a_1 a_2 \dots a_n$
MAX	$a_1 a_2 \dots a_n$	Максимальное из чисел $a_1 a_2 \dots a_n$
REM	$a b$	Остаток от деления a/b
FIX	$n1$	Возвращает целую часть числа $n1$
Тригонометрические функции		
SIN	a	$\sin(a)$, a – в радианах
COS	a	$\cos(a)$, a – в радианах
ATAN	a	$\text{arctg}(a)$ – в радианах

Имя функции	Аргументы	Возвращаемое значение
Строковые функции		
<i>STRCASE</i>	<i>s b</i>	Переводит все буквы в строке <i>s</i> в верхний (при <i>b = NIL</i>) или в нижний (при <i>b = T</i>) регистр
<i>SUBSTR</i>	<i>s n1 n2</i>	Возвращает часть строки <i>s</i> , начинающуюся с <i>n1</i> -го символа и имеющую длину <i>n2</i> символа
<i>CHR</i>	<i>n</i>	Возвращает символ с ASCII-кодом <i>n</i>
<i>STRLEN</i>	<i>s</i>	Возвращает число символов в строке <i>s</i>
<i>ASCII</i>	<i>s</i>	ASCII-код первого символа в строке <i>s</i>
Функции преобразования типов		
<i>ITOA</i>	<i>n</i>	Целое число <i>n</i> в его текстовое представление <i>s</i>
<i>ATOI</i>	<i>s</i>	Преобразует текстовую строку <i>s</i> в целое число
<i>RTOS</i>	<i>n1 n2 n3</i>	Преобразует вещественное число <i>n1</i> в текстовую строку <i>s</i> ; <i>n2</i> – код формата числа (<i>1</i> – научный; <i>2</i> – десятичный), <i>n3</i> – точность (число знаков после запятой)
<i>ATOF</i>	<i>s</i>	Преобразует текстовую строку <i>s</i> в действительное число <i>n</i>

3. Создание собственных функций. Организация диалога с пользователем

Стандартный способ создания пользовательской функции заключается в использовании функции *DEFUN*. В общем виде она записывается следующим образом:

```
(DEFUN name ( a1 a2 ... an / v1 v2 ... vm )
  ( выражение1 )
  ( выражение2 )
  ....
  ( выражение N )
  ),
```

где *name* – имя функции;

ai – *i*-й аргумент функции;

vi – *i*-я локальная переменная.

Функция *DEFUN* создает в памяти пользовательскую функцию с именем *name* и списком аргументов *a1 a2 ... an*. Для выполнения функция должна быть явно вызвана.

Пользовательские функции не могут иметь произвольное число аргументов.

Правило нахождения возвращаемого пользовательской функцией значения:

1. Найти закрывающую скобку, парную открывающей перед словом *DEFUN*, т. е. последнюю.
2. Найти предпоследнюю закрывающую скобку.
3. Найти парную ей открывающую скобку.
4. Значение, возвращаемое функцией, стоящей после этой открывающей скобки, и будет возвращаемым значением всей пользовательской функции.
5. Иначе говоря, возвращаемое значение есть результат вычисления последнего списка, находящего внутри *DEFUN*.

Пример. Введем функцию с именем *tan*, вычисляющую тангенс угла, следующего вида:

```
(DEFUN tan ( a )  
  ( / ( SIN a ) ( COS a ) )  
)
```

Все переменные в AutoLISP делятся на два вида: *глобальные* и *локальные*.

Глобальные переменные постоянно находятся в оперативной памяти и, следовательно, доступны из любой функции. Глобальные переменные создаются автоматически при присваивании им значения. Например, функция (*SETQ a 5*) создает глобальную переменную *a*.

Локальные переменные явно описываются в заголовке функции *DEFUN* и видны только внутри соответствующей пользовательской функции. В начале работы пользовательской функции локальные переменные имеют значение *NIL*. По окончании работы этой функции они автоматически удаляются из памяти.

Аргументы пользовательских функций также являются локальными переменными. В начале работы пользовательской функции аргументы принимают значения, переданные функции при ее вызове.

Пользовательская функция может не иметь как локальных переменных, так и аргументов. Поэтому возможны записи:

```
(DEFUN s  
( a b / cd )  
...)  
(DEFUN s  
( a b )  
...)  
(DEFUN s  
( / c d )  
...)  
(DEFUN s ()  
...)
```

В любом случае пара скобок после имени функции сохраняется.

Пример. Написать функцию вычисления площади треугольника со сторонами a , b , c . Воспользуемся формулой Герона. Назовем функцию *trisqu*. Ее аргументами будут стороны треугольника. Введем локальные переменную p для записи полупериметра и локальную переменную s для записи площади треугольника.

```
( DEFUN trisqu ( a b c / p s )  
  ( SETQ p ( / ( + a b c ) 2 )  
    s ( EXPT ( * p ( - p a ) ( - p b ) ( - p c ) ) 0.5 ) )  
  )
```

При вызове функции *trisqu* ей обязательно должны быть переданы три аргумента, значения которых запишутся в локальные переменные a , b , c , например: (*trisqu* 3 4 5) или (*trisqu* a 4.5 (/ 2 b)).

Рассмотрим способы занесения пользовательских функций в память и их вызова. Наиболее удобный путь – записать пользовательские функции в текстовый файл с расширением .LSP и загрузить его функцией *LOAD*, которую можно набрать в командной строке AutoCAD:

```
( LOAD "имя_файла" ),
```

или при помощи меню – Сервис/ AutoLISP / Загрузить...

Для вызова функции на исполнение принципиально возможны три способа:

1. Вызов функции с командной строки.
2. Вызов функции из меню AutoCAD (вызов функции из меню требует знания правил его модификации и не может быть рекомендован неопытным пользователям).
3. Введение новой команды в AutoCAD.

Первый способ очевиден: после загрузки файла, содержащего, к примеру, функцию вычисления тангенса *tan* достаточно просто набрать в командной строке (*tan* 0.25), чтобы узнать тангенс угла в 0.25 радиан.

Если имя пользовательской функции начинается с "C:", то после ее загрузки (и до окончания сеанса работы или до явного удаления функции из памяти) в AutoCAD появляется новая команда с именем, совпадающим с именем функции. При этом пользовательская функция не может иметь аргументов.

Чтобы просмотреть значение переменной, надо в командной строке набрать восклицательный знак, а за ним – имя переменной.

Для интерактивного запрашивания информации для функции в AutoLISP есть набор функций, обеспечивающих ввод-вывод информации.

Экран AutoCAD может работать в двух режимах: текстовом и в графическом. Эти режимы переключаются функциями AutoLISP: (*TEXTSCR*) и (*GRAPHSCR*).

Для вывода на текстовый экран предназначены следующие функции (табл. 3.1).

Таблица 3.1

Функции вывода информации на текстовый экран

Наименование	Аргументы	Описание
<i>PRINC</i>	Любое выражение	Вывод выражения на экран без учета управляющих кодов
<i>PRINI</i>	Любое выражение	Вывод выражения на экран с учетом управляющих кодов
<i>PRINT</i>	Любое выражение	Вывод выражения на экран с учетом управляющих кодов с новой строки и с пробелом в конце
<i>PROMPT</i>	Текст	Вывод текста на экран с учетом управляющих кодов
<i>TERPRI</i>	Нет	Вывод пустой строки

AutoLISP «понимает» следующие управляющие коды в выводимых на экран текстовых строках (табл. 3.2).

Таблица 3.2

Управляющие коды

Код	Значение
<i>\e</i>	Символ с кодом 27 (<i>ESC</i>)
<i>\n</i>	Переход на новую строку
<i>\r</i>	Переход в начало той же строки
<i>\t</i>	Переход на следующую позицию от табуляции (8 пробелов)
<i>\nnn</i>	Ввод символа с восьмеричным кодом <i>nnn</i>

Если нужно просто вывести на печать символ "\", его нужно удвоить, т. е. написать "\\".

Функция *PRINI* не возвращает значения и удобна для «тихого» завершения работы головной программы.

Ввод данных осуществляется семейством *GET-функций* (табл. 3.3).

Таблица 3.3

Ввод данных с клавиатуры

Наименование	Аргументы	Описание
<i>GETINT</i>	Текст подсказки	Ввод целого числа
<i>GETREAL</i>	Текст подсказки	Ввод вещественного числа
<i>GETSTRING</i>	Флаг пробела (<i>T</i> или <i>NIL</i>) Текст подсказки	Ввод текста. Если флаг пробела = <i>T</i> , в тексте могут быть пробелы, иначе (по умолчанию) пробел воспринимается как окончание ввода
<i>GETPOINT</i>	[<i>p1</i>] <i>s</i>	Ввод координат точки <i>p</i> при помощи мышки, <i>s</i> – текст подсказки. Если задана точка <i>p1</i> , то от нее к текущей точке отображается «резиновая линия»
<i>INITGET</i>	<i>n</i>	<i>NIL</i> Устанавливает защиту от неправильного ввода на одну следующую <i>GET-функцию</i> ; <i>n</i> – сумма следующих значений: 1 – запрет пустого ввода; 2 – запрет ввода нуля; 4 – запрет ввода отрицательных чисел

Все *GET-функции* возвращают введенное с клавиатуры значение или *NIL*. Для сохранения этого значения его следует записать в переменную, например:

(*SETQ m* (*GETSTRING T "\nВведите фамилию и имя:"*))

Пример. Написать интерактивную программу вычисления площади треугольника:

```
( DEFUN C:trisque ( / a b c p s )
  ( TEXTSCR )
  ( SETQ a ( GETREAL "\n Длина стороны A:" ) )
  ( SETQ b ( GETREAL "\n Длина стороны B:" ) )
  ( SETQ c ( GETREAL "\n Длина стороны C:" ) )
  ( SETQ p ( / ( + a b c ) 2 ) )
  ( SETQ s ( expt ( * p ( - p a ) ( - p b ) ( - p c ) ) 0.5 ) )
  ( PROMPT "\nПлощадь треугольника равна " )
  ( PRIN1 s ) )
```

4. Использование команд Autocad

Главная цель применения языка AutoLISP – работа с графическим экраном, что позволяет автоматизировать построение изображений.

AutoCAD имеет встроенный набор команд, который вводится с клавиатуры в ответ на приглашение «Команда:» и в большинстве запрашивают определенные параметры. Последовательность ввода этих параметров определяется форматом команды.

Большинство команд AutoCAD могут быть выполнены из программы на AutoLISP при помощи функции *COMMAND*:

(*COMMAND* *t1 t2 .. tn*),

где *t1* – имя вызываемой команды;

t2 ... tn – параметры вызываемой команды.

Есть два особых вида выражений, которые могут быть аргументами функции *COMMAND*:

- *PAUSE* позволяет пользователю ввести соответствующий параметр вручную;
- "" (две кавычки) или отсутствие параметров вообще [(*COMMAND*)] равносильно прерыванию команды.

Пример. Нарисовать из программы на AutoLISP квадрат с левым нижним углом в точке (10, 10) и стороной 25 мм. На AutoLISP это будет выглядеть так:

(*COMMAND* "ПЛИНИЯ" "10, 10" "@25, 0" "@0, 25" "@-25, 0" "ЗАМКНУТЬ").

Все константы, являющиеся параметрами функции *COMMAND*, задаются как текстовые строки, даже если они являются числами или координатами точек.

Любой параметр функции *COMMAND* можно заменить на имя переменной или выражение AutoLISP. Данный параметр примет значение, равное значению переменной или результату вычисления выражения. Но внутри функции *COMMAND* нельзя вызывать функции ввода данных (*GETREAL*, *GETSTRING* и т. д.).

Координаты точек являются списками из двух или трех вещественных чисел – координат по осям *X*, *Y* и *Z*, соответственно. Таким образом, точка с координатами 10, 10 может быть задана как текстовой строкой "10, 10", так и списком: (*LIST* 10 10).

Второй способ позволяет использовать переменные и выражения AutoLISP для указания координат.

Имеется ряд специальных встроенных функций для вычисления координат точек. Основная геометрическая функция *POLAR*:

(*POLAR a angle dist*),

где *a* – список из двух элементов (координаты точки);

angle – угол в радианах;

dist – расстояние в текущих единицах измерения.

Функция *POLAR* возвращает в виде списка координаты точки, отстоящей от точки *a* на расстояние *dist* под углом *angle*.

Положительное направление отсчета углов считается *против часовой стрелки*.

Единицы измерения, как и многие другие параметры, определяются значениями системных переменных AutoCAD.

Системная переменная – ячейка памяти, содержащая определенное значение и имеющая неизменное имя. Значения системных переменных задают различные режимы работы команд AutoCAD.

Для доступа к системным переменным в AutoLISP имеются две функции: (*GETVAR* "имя") и (*SETVAR* "имя" значение).

Функция *GETVAR* возвращает значение системной переменной с именем "имя", заданным как текстовая строка.

Например, системная переменная "*LASTPOINT*" содержит координаты текущей точки. Для ее использования в программе следует использовать функцию *GETVAR* в виде: (*GETVAR* "*LASTPOINT*").
Например:

```
( COMMAND "ПЛИНИЯ" ( LIST ( + A 10 ) ( - B 20 ) )  
( POLAR ( GETVAR "LASTPOINT" ) 0 40 ) "" )
```

Функция *SETVAR* меняет значение соответствующей системной переменной. Необходимо подумать, прежде чем менять значение системной переменной. Эти значения записываются в файл чертежа. Часть системных переменных (например, переменная, содержащая номер версии AutoCAD) доступна только для чтения и их значения нельзя изменить.

При геометрических расчетах используются также следующие функции:

- функция (*INTERS m1 m2 m3 m4 признак*) возвращает точку пересечения двух отрезков, проходящих через точки *m1* и *m2* и *m3* и *m4*, соответственно. Если точка пересечения отсутствует, функция возвращает признак = *NIL*;

- функция ($ANGLE\ m1\ m2$) возвращает угол в радианах между положительным направлением оси X и прямой, проходящей через точки $m1$ и $m2$;
- функция ($DISTANCE\ m1\ m2$) возвращает расстояние от точки $m1$ до точки $m2$ в текущих единицах измерения расстояний.

5. Работа со списками

Рассмотрим набор функций по работе со списками.

Функция ($CAR\ l$) возвращает первый элемент списка l . Если список l является описанием координат точки, то ($CAR\ l$) возвращает координату X .

Функция ($CDR\ l$) возвращает все элементы списка l , кроме первого. Иначе говоря, у списка отрывается «голова», а возвращается остающийся «хвост», при чем, даже если этот «хвост» длиной в один атом, он все равно будет списком.

В Лиспе предусмотрена возможность использования вложенных функций CAR и CDR , которые будут называться соответственно $CADR$, $CDAR$, $CAAR$, $CDDR$ и так далее (до четырех уровней вложенности). При этом ($CADR\ l$) эквивалента ($CAR\ (CDR\ l)$).

Последний элемент списка как атом возвращает функция ($LAST\ l$). В принципе ее можно использовать для получения координаты Y .

И, наконец, самая общая функция выделения элементов из списка: ($NTH\ n\ l$), которая возвращает n -й элемент списка l . Нумерация элементов списка в функции NTH начинается с нуля.

В таблице 5.1 показаны способы получения координат точки p .

Таблица 5.1

Получение координат точек

Точка p	
Координата X	Координата Y
($CAR\ p$)	($CADR\ p$)
($NTH\ 0\ p$)	($NTH\ 1\ p$)

При использовании списков, особенно списков-данных, часто необходимо добавлять в имеющийся список новые значения, как бы «приклеивая» их к его хвосту. Для этого предназначена функция ($APPEND\ l1\ l2$), которая добавляет в конец списка $l1$ список $l2$ и возвращает новый, удлиненный список. Следует обратить внимание, что для добавления в список атома его сначала нужно превратить в список из одного элемента

Пример. В переменной *a* хранится список вида (19 49). К нему нужно добавить число 20. Делается это так: (SETQ a (APPEND a (LIST 20)))

Вся эта работа со списками нужна не сама по себе, а для решения той или иной задачи. А любая сложная задача, в свою очередь, требует для своего решения применения ветвлений и циклов.

6. Управляющие конструкции AutoLISP-ветвление

Рассмотрим управляющие конструкции AutoLISP, которые в обязательном порядке содержат проверку условия. В качестве условий в AutoLISP используются логические функции, возвращающие *T* (*true* – истина) или *NIL* (ложь): "<", ">", "<=", ">=", "=", "/=".

Функции сравнения могут применяться к целым и вещественным числам, а также к текстовым строкам, но не к спискам.

Если сравниваются вещественные числа, то следует помнить об ограниченной точности вычислений с ними. Для этого следует использовать специальную функцию сравнения с заданной точностью (*EQUAL e1 e2 точность*). Здесь *точность* – число, указывающее, сколько знаков после запятой принимается во внимание при сравнении выражений *e1* и *e2*.

Функции сравнения могут объединяться при помощи логических функций, образуя сложные условия. В AutoLISP имеется четыре логические функции (табл. 6.1).

Таблица 6.1

Логические функции языка AutoLISP

<i>X</i>	<i>Y</i>	<i>X and Y</i>	<i>X or Y</i>	<i>X X or Y</i>	<i>not X</i>
<i>T</i>	<i>NIL</i>	<i>NIL</i>	<i>T</i>	<i>T</i>	<i>NIL</i>
<i>T</i>	<i>T</i>	<i>T</i>	<i>T</i>	<i>NIL</i>	<i>NIL</i>
<i>NIL</i>	<i>T</i>	<i>NIL</i>	<i>T</i>	<i>T</i>	<i>T</i>
<i>NIL</i>	<i>NIL</i>	<i>NIL</i>	<i>NIL</i>	<i>NIL</i>	<i>T</i>

Рассмотрим функцию *IF*, обеспечивающую ветвление в программе. Ее общий вид:

```
( IF c
  f1
  [f2]
 )
```


Здесь c – условие (простое или сложное); f_1 – функция, выполняемая, если условие истинно (часть «то»), а f_2 – функция, выполняемая, когда условие ложно (часть «иначе»), причем квадратные скобки говорят о том, что часть «иначе» может отсутствовать.

Пример

```
( IF ( < a 0 )  
  ( PROMPT "\nПеременная a меньше нуля" )  
  ( PROMPT "\nПеременная a больше или равна нулю" ) )
```

Если нужно в случае выполнения (или невыполнения) условия выполнить не одну операцию, а сразу несколько, то в AutoLISP используется функция (*PROGN* f_1 f_2 .. f_n). Она объединяет функции f_1 f_2 .. f_n в один блок, который можно подставить в функцию *IF*.

Пример. Посчитать действительные корни квадратного уравнения и вывести их на экран. В переменной d записан дискриминант.

```
( IF ( >= d 0 )  
  ( PROGN  
    ( SETQ x1 ( / ( + ( * b -1 ) ( SQRT d ) )  
      ( * 2 a ) ) )  
    ( SETQ x2 ( / ( - ( * b -1 ) ( SQRT d ) )  
      ( * 2 a ) ) )  
    ( PROMPT "\nX1=" )  
    ( PRINT x1 )  
    ( PROMPT "\nX2=" )  
    ( PRINT x2 )  
  ); конец PROGN  
  ( PROMPT "\nДействительных корней нет" ); "иначе"  
)
```

Функция *COND* обеспечивает множественное ветвление аналогично паскалевскому оператору *CASE*. Ее общий вид:

```
( COND  
  ( c1 f11 f12 ... f1n1 )  
  ( c2 f21 f22 ... f2n2 )  
  ...  
  ( cm fm1 fm2 ... fmnm )  
)
```

Здесь c_1 .. c_m – логические условия; f_{mn} – функции, выполняемые при выполнении того или иного условия. При чем условия проверяются последовательно до первого истинного. Если истинно сразу

несколько условий, то выполняются только функции, относящиеся к первому из них, а остальные условия даже не проверяются. Основное назначение функции *COND* – обработка ввода пользователя.

Пример

```
( SETQ a ( GETINT "\n1 – фаска, 2 – галтель, 3 – выточка" ) )  
( COND  
  ( ( = a 1 ) .... ); фаска  
  ( ( = a 2 ) .... ); галтель  
  ( ( = a 3 ) .... ); выточка  
)
```

7. Управляющие конструкции AutoLISP-циклы

Цикл выполняет двоякую функцию:

1. Выполнение одного и того же участка программы более одного раза.
2. Передача управления (под управлением понимается выполняемый в данный момент элемент программы) «назад» или «вверх» (т. е. ближе к началу программы).

Циклы в AutoLISP обеспечиваются функциями. Простейшая из них – функция *REPEAT* (*n f1 f2 ... fm*). Здесь *n* – число повторений, а *f1..fm* – те функции, которые будут выполняться *n* раз.

Назначение цикла *REPEAT* – в основном решение задач отрисовки. Например, если деталь имеет повторяющиеся элементы, можно запросить у пользователя их количество, затем вычленить повторяющийся участок и отрисовать его нужное число раз.

Пример. Нарисовать деталь – полосу, имеющую *n* отверстий, идущих с заданным шагом.

Предположим, что все данные, кроме числа отверстий, введены, сам контур детали нарисован, и левый нижний угол находится в точке с координатами (0, 0). Делаем следующее:

```
...  
( SETQ n ( GETINT "\nВведите число отверстий:" ) )  
X( LIST S ( / H 2.0 ) ) )  
( REPEAT n  
  ( COMMAND "КРУГ" X ( / D 2.0 ) )  
  ( SETQ X ( POLAR X 0 M ) )  
)
```

В этом примере в переменную *X* записывается текущая координата центра окружности; *L* – длина полосы; *H* – ширина; *S* – расстоя-

ние от левого края полосы до центра первого отверстия; D – диаметр отверстия; M – шаг отверстий.

В AutoLISP возможен цикл по условию, который задается *функцией* ((*WHILE* c f_1 f_2 ... f_m). Здесь c – логическое условие. Цикл выполняется, пока это условие истинно.

Пусть в той же детали, рассмотренной выше, сверлятся отверстия, причем центр первого из них отстоит от левого края на величину S , а край последнего должен отстоять от правого края на величину B .

В нашем случае p_0 – координата левого нижнего угла детали, а x координата правого конца детали:

```
( SETQ pr ( + ( CAR p0 ) L ) )  
; Координата центра первого отверстия  
( SETQ x ( + ( CAR p0 ) S ) )  
( WHILE ( < ( - xr ( + x ( / D 2 ) ) ) B )  
  ( COMMAND "КРУГ"  
    ( LIST x ( + ( CADR p0 ) H ) ) D )  
  ( SETQ x ( + x m ) )  
)
```

При использовании цикла «пока» нужно позаботиться о задании начальных условий.

В AutoLISP для обработки каждого элемента списка предусмотрены два особых цикла: *FOREACH* и *MAPCAR*.

Рассмотрим цикл (*FOREACH* $name$ $list$ exp). Здесь $name$ – имя переменной, в которую последовательно записываются элементы списка $list$, а exp – выражение AutoLISP, выполняемое столько раз, сколько элементов есть в списке $list$.

Рассмотрим работу цикла на примере. Пусть нужно просуммировать все элементы списка $data$ и записать сумму в переменную s .

```
; Обнуление суммы  
( SETQ s 0 )  
( FOREACH cur data ( SETQ s ( + s cur ) ) )
```

При выполнении этого фрагмента происходит следующее: берется список $data$, по очереди каждый его элемент, начиная с первого, записывается в переменную cur , после чего выполняется выражение

```
( SETQ s ( + s cur ) )
```

В AutoLISP нет возможности изменить значение отдельного элемента списка. Например, при масштабировании изображения нуж-

но все элементы списка размеров *size* умножить на масштабный коэффициент *k*.

Это можно сделать при помощи функции (*MAPCAR 'f l1 l2 ... ln*).

MAPCAR выполняет функцию *f* поочередно над первыми, вторыми, третьими... элементами списков *l1 .. ln*. При этом *n* должно быть равно числу аргументов функции *f*, а все списки *l1 ... ln* должны содержать одинаковое число элементов. Самое главное – функция возвращает список результатов выполнения функции *f*. Перед функцией нужно ставить апостроф, потому что *MAPCAR* нужна просто ссылка на то место в памяти, где эта функция находится, а не результат выполнения этой функции.

Пример. Все элементы списка размеров *size* нужно умножить на масштабный коэффициент *k*. Функция умножения требует как минимум два аргумента, в данном случае – текущий элемент списка и коэффициент *k*. Придется сформировать вспомогательный список, состоящий из стольких значений *k*, сколько есть размеров в списке *size*. Делается это так.

```
( SETQ coeff NIL )  
( REPEAT ( LENGTH size )  
  ( SETQ coeff ( APPEND coeff ( LIST k ) ) )  
)
```

После этого можно применить функцию *MAPCAR*:

```
( SETQ size ( MAPCAR '* size coeff ) )
```

При таком решении приходится создавать лишний список *coeff*.

Было разработано так называемое λ -исчисление Черча. Само по себе оно представляет математический аппарат для описания функций. В AutoLISP λ -исчисление заключается в наличии возможности создать «одноразовую» функцию, даже не имеющую своего имени. Причем внутри этой функции будут видны все текущие локальные переменные. «Одноразовая» функция создается функцией (*LAMBDA (arg1 ... argn) exp1 ... expm*). Здесь *arg1 ... argn* – список аргументов «одноразовой» функции, а *exp1 ... expm* – выражения AutoLISP, выполняющие какие-то операции над аргументами.

Функцию *LAMBDA* можно прямо записать как аргумент функции *MAPCAR*, не забыв поставить апостроф для подавления ее выполнения:

```
( MAPCAR
  '( LAMBDA ( x ) ( * x k ) )
  size
)
```

Здесь определена функция с одним аргументом x , которая вычисляется для каждого элемента списка $size$. Полученные произведения «склеиваются» в список, являющийся возвращаемым значением функции *MAPCAR*.

8. Основы параметрического проектирования

Следует определить, какие изделия можно считать подлежащими параметризации. При рассмотрении 2D-проекций детали видно, что они могут быть разбиты на элементарные графические примитивы: отрезки и дуги. Каждый примитив однозначно определяется координатами своих базовых точек: начальной и конечной точек отрезка, начальной, конечной точек и центра дуги. Тогда проекцию можно представить в виде графа, вершины которого соответствуют базовым точкам, а ребра – параметрическим связям между ними.

Каждая связь $i-j$, проходящая от i -й до j -й базовой точки, есть вектор параметров

$$(\overline{d_{ij}}, \alpha_{ij}),$$

где d_{ij} – расстояние от точки i до точки j ;

α_{ij} – угол между прямой, проходящей через точки i и j , и прямой, выбранной в качестве начала отсчета углов.

Два объекта называются конструктивно подобными, если их соответствующие проекции представляются одними и теми же графами.

Использование графов дает возможность, задавшись произвольными координатами x_i, y_i i -й базовой точки, однозначно определить координаты всех остальных базовых точек при обходе графа по формулам:

$$\left. \begin{aligned} x_j &= x_i + d_{ij} \cdot \cos \alpha_{ij} \\ y_j &= y_i + d_{ij} \cdot \sin \alpha_{ij} \end{aligned} \right\}$$

Таким образом, имея граф, описывающий семейство однотипных объектов, конструктору достаточно задать размерные связи между его базовыми точками, а специализированная САПР выполнит об-

ход графа, расчет координат и отображение полученной проекции. Для этого представим граф в виде функции отображения (топологической функции) вида

$$\mathfrak{R}(x, y, \bar{d}),$$

где x, y – координаты начальной точки;

\bar{d} – вектор параметров графа.

Многие изделия представляются в виде вариантных чертежей, когда изделие состоит из постоянной части с варьируемыми размерами и вариантной части с уникальной геометрией. Например, для станочных тисков проектируются уникальные губки под каждую конкретную деталь, фиксируемую этими тисками, но корпус, ходовой винт, струбцина и прочие ДСЕ тисков остаются конструктивно неизменными. В данном случае определение конструктивно подобного изделия не выполняется. Поэтому необходимо ввести понятие варианта конструкции, для чего следует разбить граф проекции на константную часть C и переменную часть V . Тогда достаточно потребовать соответствия графов только константных частей для их автоматизированного параметрического проектирования, а варианты части, являющиеся уникальными, проектируются с применением универсальных САПР с последующим объединением частей C и V .

Очень часто конструктору приходится выпускать документацию на ряд изделий, которые отличаются только своими размерами (линейными или угловыми), а форма их остается неизменной. Пример таких изделий – технологическая оснастка: кондукторы, мерительный инструмент, пресс-формы и др.

Сущность *параметрического проектирования* состоит в создании математической модели класса конструктивно однородных изделий, а затем в генерации изображений этих изделий по набору задаваемых размерных параметров.

При параметрическом проектировании конструктор запускает программу, рассчитанную на определенный класс изделий, и вводит требуемые размеры. Программа отрисовывает на экране чертеж детали. Конструктор оценивает его и при необходимости вводит размеры снова до достижения требуемого результата. Одновременно может рассчитываться масса детали, что позволяет контролировать ее «на ходу», прямо в процессе проектирования.

Проекция изделия как векторное изображение состоит из множества *базовых геометрических элементов* – отрезков и дуг. Положение этих элементов на плоскости определяется координатами их ба-

зовых точек. Для отрезка базовыми точками являются его начало и конец, а для дуги – начало, конец и центр (дугу можно задать и через другие параметры: радиус, угол, направление и т. д.).

Таким образом, программа-параметризатор работает по следующему алгоритму:

1. Ввод исходных данных.
2. Отрисовка текущего варианта.
3. Запрос пользователю: повторить?
4. Если да, то переход на п. 1.
5. Конец.

Пример. Написать простейший параметризатор для детали, изображенной на рис. 8.1.

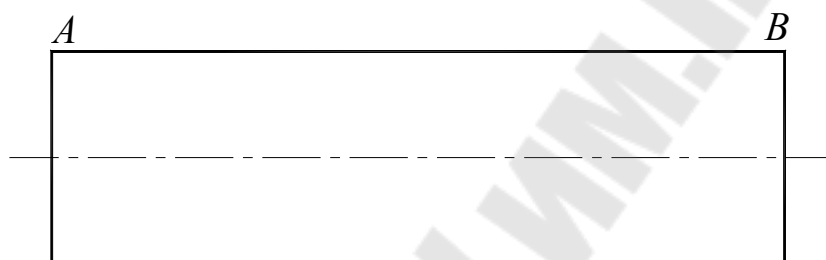


Рис. 8.1. Заданная геометрия детали

Сначала нужно определиться с координатами базовых точек. Поскольку проекция симметрична, достаточно найти координаты только двух точек A и B . Нижняя половина проекции отрисовуется автоматически при помощи команды AutoCAD «ЗЕРКАЛО».

Размерными параметрами детали будут длина цилиндра и его диаметр. Условимся считать опорной точкой, т. е. определяющей положение проекции на листе чертежа, левый конец осевой линии. Согласно ЕСКД осевая линия должна выходить за контур на 5 мм.

На входе программы-параметризатора будут координата X точки привязки; координата Y точки привязки; длина цилиндра L ; диаметр цилиндра D . Требуется найти координату X точки A (обозначим A_X); координату Y точки A (обозначим A_Y); координату X точки B (обозначим B_X); координату Y точки B (обозначим B_Y).

Очевидно, что:

$$A_X = X + 5; \quad B_X = X + 5 + L;$$

$$A_Y = Y + \frac{D}{2}; \quad B_Y = Y + \frac{D}{2}.$$

Будем считать, что функция отображения *SHOW* вызывается из головной функции и ей передаются параметры X, Y, L, D . Результаты расчета нужно записывать в локальные переменные функции *SHOW*. В переменную *tmp* заносятся текущие результаты расчетов, а в переменную *lst* – список рассчитанных координат. Этот список будет состоять из двух элементов (точки *A* и *B*), каждый из которых, в свою очередь, тоже будет списком их двух координат: X и Y :

$$lst \leftarrow \left(\underbrace{(A_X \ A_Y)}_{\text{точка } A} \underbrace{(B_X \ B_Y)}_{\text{точка } B} \right).$$

Ниже представлена программа функции *SHOW* (расчет координат базовых точек и отрисовки).

```
( DEFUN show ( x y d l / tmp lst )
; x, y, s, l – параметры
; tmp, lst – локальные переменные
( SETQ tmp ( + x 5 ) ); координата x точки A записывается в переменную tmp
( SETQ lst ( LIST tmp ( + y ( / d 2 ) ) ) ); в переменную lst записывается список координат точки A
; координата x точки B записывается в переменную tmp
( SETQ tmp ( + x 5 1 ) ); в переменную tmp записывается список координат точки B
( SETQ tmp ( LIST tmp ( + y ( / d 2 ) ) ) ); в список lst добавляется точка B
( SETQ lst ( LIST lst tmp ) )
( COMMAND "СТЕРЕТЬ" "РАМКА" '( -10000 -10000 ) '(10000 10000) "" ); Удаляем все объекты
; Рисуем осевую линию красным цветом (код 1)
; и штрих-пунктирно (тип линии CENTER)
( COMMAND "ЦВЕТ" 1 "ТИПЛИН" "УСТАНОВИТЬ" "ОСЕВАЯ2" "" )
( COMMAND "ПЛИНИЯ" ( LIST x y ) ( LIST ( + x 1 10 ) y ) "" )
; Рисуем контур детали белым цветом (код 1)
; и сплошной линией (тип линии CONTINUOUS)
( COMMAND "ЦВЕТ" 7 "ТИПЛИН" "УСТАНОВИТЬ" "CONTINUOUS" "" )
( COMMAND "ПЛИНИЯ"
( LIST ( + x 5 ) y ) ; левая точка пересечения осевой с контуром
( CAR lst ) ; точка A
( CADR lst ) ; точка B
```



```

( LIST ( + x l 5 ) y ) ; правая точка пересечения осевой с контуром
"" )
( COMMAND "ЗЕРКАЛО" ( CAR lst ) "" ( LIST xy ) ( LIST ( + x l ) y ) "H" )
( COMMAND "ЦВЕТ" 1 )
( COMMAND "РАЗМЕР1"
"ГОР" ; размер горизонтальный
( CAR lst ) ; начало первой выносной линии – точка A
( CADR lst ) ; начало второй выносной линии – точка B
( POLAR ( CAR lst ) ( / PI 2 ) 30 ) ; размерная линия
; отстоит от контура на 30 мм
"" ) ; размерный текст ставится автоматически
; Проставляем диаметр
( COMMAND "РАЗМЕР1" "ВЕР" ( CADR lst )
( LIST ( CAR ( CADR lst ) ) ( – ( CADR ( CADR lst ) ) d ) )
( POLAR ( CADR lst ) 0 30 )
( STRCAT "%%c" ( RTOS d 2 2 ) ) )
( COMMAND "ПОКАЖИ" "ВСЕ" )
); конец функции

```

Размерные параметры конкретного изделия вводятся конструктором с клавиатуры. Функция ввода данных *GETDIM* на входе не имеет параметров, а возвращает несколько чисел, объединенных в список:

```

( DEFUN getdim ( / l d )
; ввод длины и диаметра
( INITGET 7 ) ; запрет пустого ввода и ввода чисел <=0
( SETQ l ( GETREAL "\nВведите длину валика: " ) )
( INITGET 7 )
( SETQ d ( GETREAL "\nВведите диаметр валика: " ) )
( LIST d l ) ; возвращаемое значение
); конец функции getdim

```

Головная функция выполняет подготовительные операции и циклический вызов двух остальных функций: *GETDIM* и *SHOW*. Для удобства перед именем головной функции добавлены символы "C:". Тогда в AutoCAD появится новая команда с именем головной функции и можно будет не набирать скобки при ее вызове. В примере головная функция называется *C:MAIN*, поэтому для ее вызова в командной строке AutoCAD достаточно набрать *MAIN*:

```

( DEFUN C:main ( / l x y flag )
; Устанавливаем размерные переменные по ЕСКД
( SETVAR "DIMTAD" 1 ); Текст над размерной линией, а не в разрыве
( SETVAR "DIMTOH" 0 ); Текст вне размерных линий параллелен
линиям
( SETVAR "DIMTIH" 0 ); Текст между размерными линиями
;горизонтален
( SETVAR "DIMTOFL" 1 ); Проведение линии между выносными,
если
;текст сбоку
( SETVAR "DIMEXE" 1 ); Продолжение выносных линий за размер-
ными
( SETVAR "DIMTOL" 0 ); Отключение генерации допусков
( SETVAR "DIMASZ" 3 ); Размер стрелок
( SETVAR "DIMEXE" 3 ); Продолжение выносных линий за размер-
ную
( SETQ flag T )
( WHILE flag
( SETQ l ( getdim ) x 10 y 10 )
( show x y ( CAR l ) ( CADR l ) )
( SETQ ans ( GETSTRING "\nПовторить?<Д/Н>: " ) )
( SETQ flag ( OR ( = ans "Д" ) ( = ans "д" ) ) )
); конец WHILE
( PRIN1 )
)

```

Пример. Создание параметрического изображения многоступенчатых объектов.

Требуется создать различные функции на языке AutoLISP для построения изображения многоступенчатого объекта (колонны, ракеты, валов и т. д.) с заданным числом ступеней (N) и размерами каждой ступени: ширина (диаметр) ступени – D ; длина ступени – L .

Программы изображения многоступенчатого объекта с размерами включают следующие функции:

- ($MNOGOSTR X Y LD$) – функция создания параметрического изображения многоступенчатого объекта и указания его размеров. Аргументы функции: X – координата x базовой точки ступени; Y – координата y базовой точки ступени; LD – список пар диаметров и длин ступеней объекта '($D1 L1$)'($D2 L2$)'($D3 L3$) ...);

- (*SETDIM RMAX*) – функция установки размерных переменных для определения вида размерных линий и самих размеров. *RMAX* – максимальный из всех радиусов ступеней объекта;

- (*RAZMV*) – функция указания размеров элементов чертежа;

- (*STUP*) – функция создания изображения ступени объекта.

Пример вызова функции (*MNOGOSTR X Y LD*) –

(*MNOGOSTR '10 100 '((40 70) (60 50) (40 80) (70 100))*);

(*DEFUN SETDIM (RMAX)*

(*SETVAR "DIMTAD" 1*) ;Текст над размерной линией.

(*SETVAR "DIMSOXD" 1*) ;Текст между выносными линиями.

(*SETVAR "DIMTIH" 0*) ;Текст расположен параллельно линии.

(*SETVAR "DIMDLI" (* RMAX 0.25)*) ; Отступ между линиями.

(*SETVAR "DIMEXE" (* RMAX 0.05)*)

;Удлинение размерной линии за выносную.

(*SETVAR "DIMTSZ" 0*) ;Изображение стрелки на концах размерной линии.

(*SETVAR "DIMASZ" (* RMAX 0.15)*) ;Длина стрелки

)

(*DEFUN RAZMV ()*

(*SETDIM RMAX*); Определение вида размерных линий

(*SETQ LT2T3 (REVERSE LT2T3)*)

(*COMMAND "СТИЛЬ" "" "TXT" (* RMAX 0.15) "1" "" "" "" ""*)

(*FOREACH EL LT2T3* ;Цикл ввода размеров диаметров ступеней

(*SETQ T2 (CAR EL)* ;Точка ввода первой выносной линии

T3 (CADR EL) ;Точка ввода второй выносной линии

TR (LIST (- (CAR T2) 10) (CADR T2)))

(*COMMAND "РАЗМЕР1" "ВЕРТИКАЛЬНЫЙ" T2 T3 TR "%%с<>"*)

)

(*SETQ TR (LIST X (- Y (* RMAX 1.2))*)

I 0)

(*FOREACH EL LT2T3* ;цикл указания длины ступеней

(*SETQ T3 (CADR EL)* ;точка ввода второй выносной

I (+ I 1))

(*IF (= I 1)*

```

(COMMAND "РАЗМЕР1" "ГОРИЗОНТАЛЬНЫЙ" BP T3 TR
"" )
(COMMAND "РАЗМЕР1" "БАЗОВЫЙ" T3 "" )
)
)
(DEFUN STUP ()
  (SETQ T10 ( POLAR BT (/ PI 2) R0)
    T1 ( POLAR BT (/ PI 2) R)
    T2 (POLAR T1 0 L)
    T3 (POLAR T2 (* 1.5 PI) D)
    T4 ( POLAR T3 PI L)
    T40 ( POLAR BT (* 1.5 PI) R0)
    BT ( POLAR BT 0 L))
  (COMMAND "ПЛИНИЯ" T10 "ширина" 1 1 T1 T2 T3 T4 T40 "" )
)
(DEFUN MNOGOSTR(X Y LD) ;начало создания функции
  (SETVAR "CMDECHO" 0) ;Отключение эха команд.
;УСТАНОВКА ФОРМАТА ЧЕРТЕЖА
  (COMMAND "ЛИМИТЫ" "0,0" "297,210")
  (COMMAND "ПОКАЗАТЬ" "границы")
  (TEXTSCR) ;Переход на текстовый режим.
  (SETQ RMAX 0
    LT2T3 '()
    R0 0
    BT (LIST X Y)
    BP BT)
  (FOREACH EL LD ; Начало цикла создания ступени
    (SETQ D ( CAR EL) ; диаметра ступени объекта,
      L (CADR EL );длины ступени объекта.
      R (* D 0.5 ));Определение радиуса ступени
    ( IF (> R RMAX) (SETQ RMAX R))
    (IF (< R R0) (SETQ R0 R));Определение R0 для первой ступени
    (STUP) ;Изображение первой ступени
    (SETQ LT2T3 (CONS (LIST T2 T3) LT2T3)
; список пар для изображения выносных линий
    R0 R))
  (RAZMV)
  PRIN1

```

```
(SETQ BP ( LIST (- (CAR BP) 5) (CADR BP) )
      BT ( LIST (+ (CAR BT) 5) (CADR BT) ))
(COMMAND "ТИПЛИН" "УСТАНОВИТЬ" "ОСЕВАЯ2" "")
(COMMAND "ЦВЕТ" 1 ""
      "ОТРЕЗОК" BP BT "")
(COMMAND "ТИПЛИН" "УСТАНОВИТЬ" " CONTINUOUS" "")
(COMMAND "ЦВЕТ" 7 "")
)
(MNOGOSTR '10 100 '((40 70) (60 50) (40 80) (70 100)))
```

Пример 3. Разработать программу размещения текстовых данных (например, технические требования) из файла на чертеже.

Ниже следует функция (*TEXTIN1*), которая обеспечивает считывание заданного текстового файла построчно. Полное имя файла запоминается в локальной переменной *FN*. В локальной переменной *BP* сохраняется начальная точка первой строки (записи) текста, в локальной переменной *P* – высота букв текста, *U* – угол наклона текста. В локальной переменной *OF* хранится полное имя файла, открытого для чтения «*r*» (*Read*), в локальную переменную *RL* – считывается строка.

```
(DEFUN TEXTIN1 (/ FN BP OF RL P U)
  (SETVAR "CMDECHO" 0)
  ;(SETVAR "TEXTSCR")
  (COMMAND "ЛИМИТЫ" "0,0" "210,297")
  (COMMAND "ПОКАЗАТЬ" "ГРАНИЦЫ")
  (SETQ FN (GETSTRING "\n Введите имя файла с текстом:" )
      BP (GETPOINT "\n Введите точку ввода текста : [можно мыш-
кой]")
      P (GETREAL "\n Введите высоту букв текста:")
      U (GETREAL "\n Введите угол наклона текста:"))
  (TYPE FN)
  (PRINT FN)
  (SETQ OF (OPEN FN "r")
      H (/ P 4)
      RL (READ-LINE OF ))
  (COMMAND "СТИЛЬ" "" "" P H "" "" "" "")
  (COMMAND "ТЕКСТ" BP U RL)
```

```
(WHILE (SETQ RL (READ-LINE OF ))
(SETQ BP (LIST (CAR BP) (- (CADR BP) 10)))
(COMMAND "ТЕКСТ" BP U RL) )
(CLOSE OF)
)
```

9. Задания к лабораторным работам

9.1. Разработка программы-параметризатора

1. Выбрать из таблицы 9.1 вариант параметрической модели, указанный преподавателем.

2. Установить набор параметров s_1, s_2, \dots, s_n , определяющих параметрическую модель изделия. Задать точку привязки a .

3. Написать функцию ввода данных: вход – ничего, выход – список вида: $(a s_1 \dots s_n)$, где a , в свою очередь, список двух координат точки привязки. Функция выполняет ввод данных с клавиатуры. Должна проводиться проверка вводимых размеров на неотрицательность.

4. Написать функцию отрисовки: вход – список $(a s_1 \dots s_n)$, выход – ничего. Функция выполняет отрисовку чертежа детали. Перед отрисовкой экран необходимо очистить. Нужно устанавливать требуемые цвет и тип линий, а после отрисовки выполнить команду «Покажи Все». Должны быть проставлены все параметризуемые размеры.

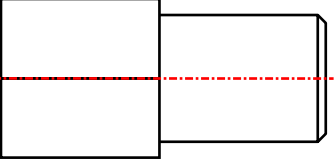
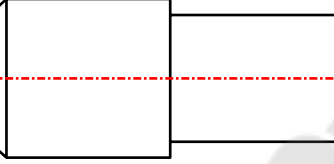
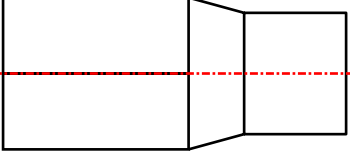
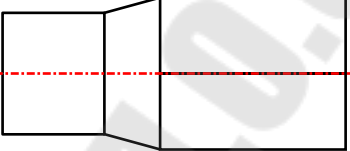
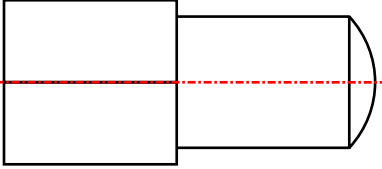
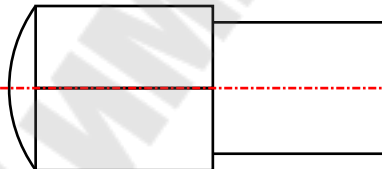
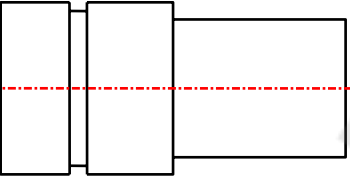
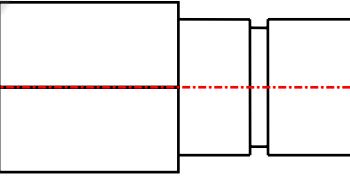
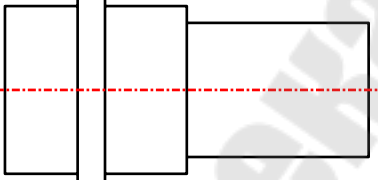
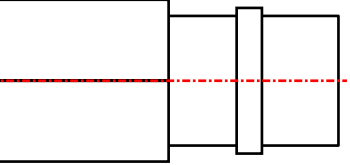
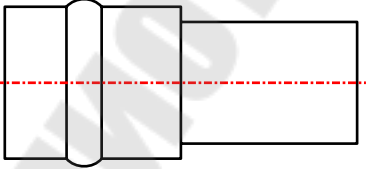
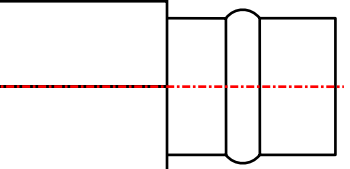
5. Написать головную функцию. В ней в цикле выполняются следующие действия:

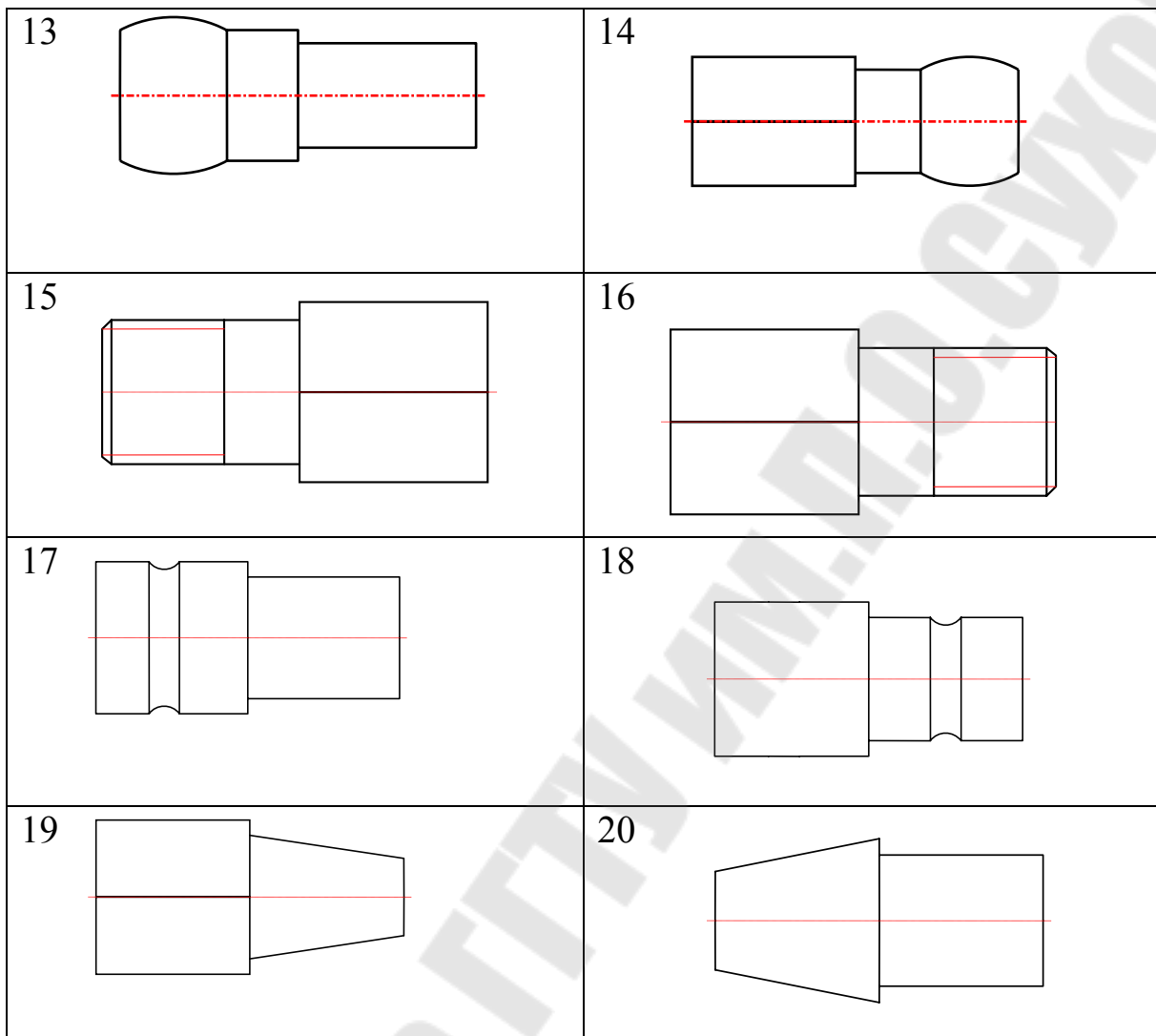
- инициализация (например, установка размерных переменных);
- вызов функции ввода данных и запоминание возвращаемого ей списка в локальной переменной;
- вызов функции отрисовки с передачей ей в качестве параметра списка, возвращенного функцией ввода данных;
- запрос пользователю: «Продолжить? <Да/Нет>» В зависимости от ответа либо прекратить выполнение программы (с «тихим выходом»), либо повторить ввод размеров, отрисовку и т. д. (цикл).

6. Написать функцию вычисления массы детали: вход – список $(s_1 \dots s_n)$, выход – масса детали данных размеров в кг (плотность стали $7,85 \text{ г/см}^3$).

7. Выводить массу детали на экран после отрисовки.

Варианты заданий

1 	2 
3 	4 
5 	6 
7 	8 
9 	10 
11 	12 



9.2. Отработка навыков программирования размерных задач на языке AutoLISP (варианты задач)

1. Написать функцию, запрашивающую у пользователя два целых числа и выводящую на экран большее из них.
2. Написать функцию, запрашивающую у пользователя два целых числа и выводящую на экран меньшее из них.
3. Написать функцию, запрашивающую у пользователя два целых числа a и b и выводящую на экран текст «Числа равны», « A больше B », « B больше A ».
4. Написать функцию, запрашивающую у пользователя сторону квадрата и отрисовывающую квадрат с заданной длиной стороны и левым нижним углом в точке $(10, 10)$.

5. Написать функцию, запрашивающую у пользователя сторону треугольника и отрисовывающую равносторонний треугольник с заданной длиной стороны и левым нижним углом в точке (5, 0).
6. Написать функцию, запрашивающую у пользователя координаты концов двух отрезков и выводящую текст «Отрезки пересекаются» или «Отрезки не пересекаются».
7. Написать функцию, запрашивающую длину и толщину отрезка и отрисовывающую отрезок с началом в точке (0, 0) параллельно оси OX .
8. Написать функцию, запрашивающую длину и толщину отрезка и отрисовывающую отрезок с началом в точке (0,0) параллельно оси OY .
9. Написать функцию, отрисовывающую правильный пятиугольник с длиной стороны 10 и координатами центра описанной окружности (100, 100).
10. Написать функцию, запрашивающую имя пользователя и выводящую текст «Привет, <имя>!».
11. Написать функцию, запрашивающую у пользователя два числа и выводящую на экран значение функции $y = \sin(a + b) - \text{tg}(a / b)$.
12. Написать функцию, запрашивающую у пользователя два числа и выводящую на экран значение функции $y = |a - \text{tg}b|$.
13. Написать функцию, запрашивающую у пользователя два числа и выводящую на экран значение функции $y = \ln|a - b|$.
14. Написать функцию, запрашивающую у пользователя два числа и выводящую на экран значение функции $y = \sqrt{a + b^2}$.
15. Написать функцию *COPYL*, которая строит копию списка L .
16. Написать функцию *REMOVE*, которая удаляет из списка L все совпадающие с данным символом A (списком) элементы.
17. Написать функцию *EARLYAB*, которая проверяет, находится ли элемент A ранее элемента B в списке LU .
18. Написать функцию *ADDNL*, которая прибавляет к числовым элементам списка L число K .
19. Написать функцию *LINEZ*, которая изображает горизонтальную линию с заданным числом выбранного знака.
20. Написать функцию *DELLAST*, которая удаляет последний элемент списка.

Литература

1. Бугрименко, Г. А. Автолисп – язык графического программирования в системе AutoCAD / Г. А. Бугрименко. – Москва : Машиностроение, 1992. – 144 с.
2. Кудрявцев, Е. М. AutoLISP. Основы программирования в AutoCAD 2000 / Е. М. Кудрявцев. – Москва : ДМК Пресс, 2000. – 416 с.
3. Хювенен, Э. Мир Лиспа. В 2 т. / Э. Хювенен, Й. Сеппанен. – Москва : Мир, 1985.

Содержание

Введение.....	3
1. Классификация языков программирования. Язык AutoLISP	4
2. Присваивание значений в AutoLISP. Встроенные функции.....	6
3. Создание собственных функций. Организация диалога с пользователем.....	8
4. Использование команд Autocad	13
5. Работа со списками	15
6. Управляющие конструкции AutoLISP-ветвление	16
7. Управляющие конструкции AutoLISP-циклы	18
8. Основы параметрического проектирования	21
9. Задания к лабораторным работам	30
Литература	34

Учебное электронное издание комбинированного распространения

Учебное издание

ИСПОЛЬЗОВАНИЕ ЯЗЫКА AUTOLISP ДЛЯ АВТОМАТИЗИРОВАННОГО ПРОЕКТИРОВАНИЯ

**Лабораторный практикум
по курсу «Основы автоматизированного
проектирования» для студентов специальностей
1-36 01 01 «Технология машиностроения»
и 1-36 01 03 «Технологическое оборудование
машиностроительного производства»
дневной и заочной форм обучения**

Электронный аналог печатного издания

Автор-составитель: **Мурашко** Валентина Семеновна

Редактор *Л. Ф. Теплякова*
Компьютерная верстка *Н. Б. Козловская*

Подписано в печать 23.04.07.

Формат 60x84/16. Бумага офсетная. Гарнитура «Таймс».

Цифровая печать. Усл. печ. л. 2,32. Уч.-изд. л. 2,20.

Изд. № 34.

E-mail: ic@gstu.gomel.by

<http://www.gstu.gomel.by>

Издатель и полиграфическое исполнение:
Издательский центр учреждения образования
«Гомельский государственный технический университет
имени П. О. Сухого».

ЛИ № 02330/0131916 от 30.04.2004 г.

246746, г. Гомель, пр. Октября, 48.