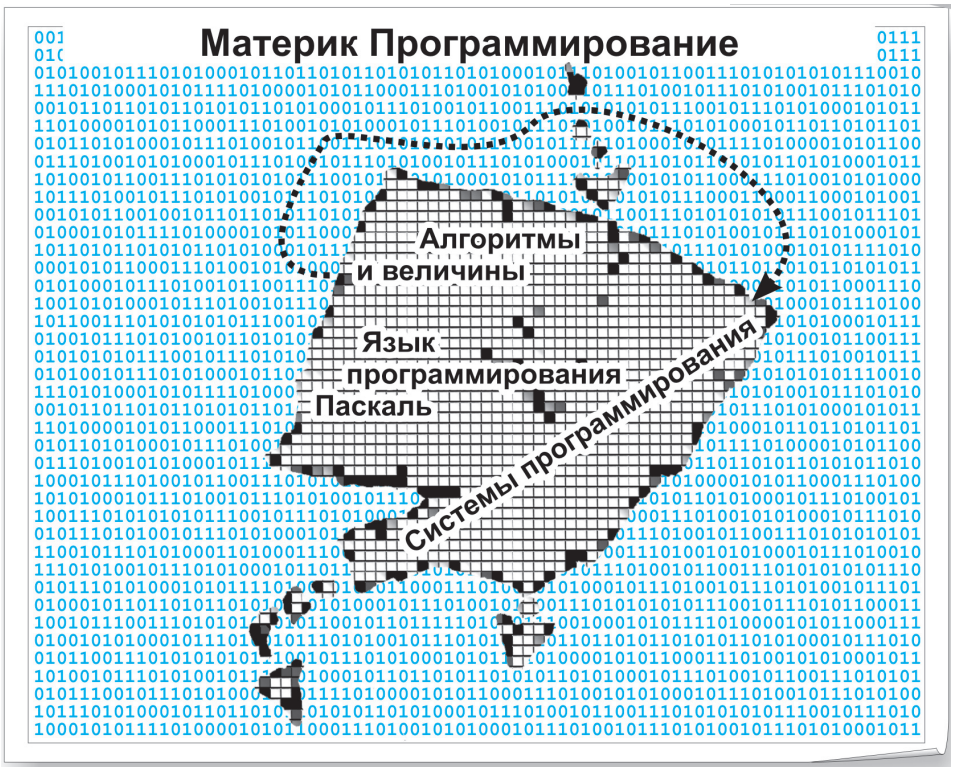
Глава II
ВВЕДЕНИЕ В ПРОГРАММИРОВАНИЕ



Здесь вы узнаете:

- что такое программирование
- как строятся вычислительные алгоритмы
- как составляются программы на языке Паскаль

§ 8
Что такое программирование

Основные темы параграфа:

- *кто такие программисты*;
- *что такое язык программирования*;
- *что такое система программирования.*

Кто такие программисты

Теперь вам предстоит ближе познакомиться еще с одним разделом информатики, который называется «Программирование».

Назначение **программирования** — разработка программ управления компьютером с целью решения различных информационных задач.

Специалисты, профессионально занимающиеся программированием, называются **программистами**. В первые годы существования ЭВМ для использования компьютера в любой области нужно было уметь программировать. В 1970–1980-х годах начинает развиваться прикладное программное обеспечение. Бурное распространение прикладного ПО произошло с появлением персональных компьютеров. Стало совсем не обязательным уметь программировать для того, чтобы воспользоваться компьютером. Люди, работающие на компьютерах, разделились на **пользователей** и **программистов**. В настоящее время пользователей гораздо больше, чем программистов.

Может возникнуть впечатление, что программисты теперь уже и не нужны! Но кто же тогда будет создавать все операционные системы, редакторы, графические пакеты, компьютерные игры и многое другое? Программисты, безусловно, нужны, причем задачи, которые им приходится решать, со временем становятся все сложнее.

Программирование принято разделять на системное и прикладное. **Системные программисты** занимаются разработкой системного программного обеспечения: операционных систем, утилит и пр., а также систем программирования. **Прикладные программисты** создают прикладные программы: редакторы, табличные процессоры, игры, обучающие программы и др. Спрос на высококвалифицированных программистов, как системных, так и прикладных, очень большой.

В данной главе вы познакомитесь с простейшими правилами и приемами программирования, заглянете в эту актуальную и престижную профессиональную область.

Что такое язык программирования

Для составления программ существуют разнообразные **языки программирования.**

Язык программирования — это фиксированная система обозначений для описания алгоритмов и структур данных.

За годы существования ЭВМ было создано много языков программирования. Наиболее известные среди них: Фортран, Паскаль, Бейсик, С (Си) и др. Распространенными языками программирования сегодня являются C++, Java, Pascal, Basic, Python.

Что такое система программирования

Для создания и исполнения на компьютере программы, написанной на языке программирования, используются **системы программирования.**

Система программирования — это программное обеспечение компьютера, предназначенное для разработки, отладки и исполнения программ, записанных на определенном языке программирования.

Существуют системы программирования на Паскале, Бейсике и других языках.

В данной главе речь будет идти о средствах и способах универсального программирования — не ориентированного на какую-то узкую прикладную область. Примером узкоспециализированного программирования является веб-программирование, ориентированное на создание веб-сайтов. Для этих целей, например, используется язык JavaScript. Языки Паскаль, Бейсик, Си относятся к числу **универсальных языков программирования.**

Разработка любой программы начинается с построения алгоритма решения задачи. Ниже мы обсудим особенности алгоритмов решения задач обработки информации на компьютере.

Коротко о главном

Программирование — область информатики, посвященная разработке программ управления компьютером с целью решения различных информационных задач.

Программирование бывает системным и прикладным.

Паскаль, Бейсик, Си, Фортран — это универсальные языки программирования.

Система программирования — это программное обеспечение компьютера, предназначенное для разработки, отладки и исполнения программ, записанных на определенном языке программирования.

Вопросы и задания

1. Что такое программирование?
2. Какие задачи решают системные и прикладные программисты?
3. Назовите наиболее распространенные языки программирования.
4. В чем состоит назначение систем программирования?

www ЕК ЦОР: часть 2, глава 6, § 32. ЦОР № 2, 5.

§ 9
Алгоритмы работы с величинами

Основные темы параграфа:

- *компьютер как исполнитель алгоритмов*;
- *величины*: *константы и переменные*;
- *система команд*;
- *команда присваивания*;
- *команда ввода*;
- *команда вывода*.

Компьютер как исполнитель алгоритмов

Вам уже известно, что всякий алгоритм составляется для конкретного исполнителя. *Теперь в качестве исполнителя мы будем рассматривать компьютер, оснащенный системой программирования на определенном языке.*

Компьютер-исполнитель работает с определенными *данными* по определенной *программе*. Данные — это множество величин.

Величины: константы и переменные

Компьютер работает с информацией, хранящейся в его памяти. Отдельный информационный объект (число, символ, строка, таблица и пр.) называется величиной.

Всякая обрабатываемая программой величина занимает свое место (поле) в памяти компьютера. Значение величины — это информация, хранимая в этом поле памяти.

Существуют *три основных типа величин*, с которыми работает компьютер: **числовой**, **символьный** и **логический**. Изучая базы данных и электронные таблицы, вы уже встречались с этими типами. В данной главе мы будем строить алгоритмы, работающие с числовыми величинами.

Числовые величины в программировании, так же как и математические величины, делятся на переменные и константы (постоянные). Например, в формуле $(a^2 - 2ab + b^2)$ a, b — переменные, 2 — константа.

Константы записываются в алгоритмах своими десятичными значениями, например: 23, 3.5, 34. Значение константы хранится в выделенной под нее ячейке памяти и остается неизменным в течение работы программы.

Переменные в программировании, как и в математике, обозначаются символическими именами. Эти имена называют **идентификаторами** (от глагола «идентифицировать», что значит «обозначать», «символизировать»). Идентификатор может быть одной буквой, множеством букв, сочетанием букв и цифр и т. д. Примеры идентификаторов: *A*, *X*, *B*3, *prim*, *r*25 и т. п.

Система команд

Вам известно, что всякий алгоритм строится исходя из системы команд исполнителя, для которого он предназначен. Любой алгоритм работы с величинами может быть составлен из следующих команд:

- присваивание;
- ввод;
- вывод;
- обращение к вспомогательному алгоритму;
- цикл;
- ветвление.

Эти команды существуют во всех языках, поддерживающих структурное программирование: в Паскале, Си и др.

Команда присваивания

Команда присваивания — одна из основных команд в алгоритмах работы с величинами. Записывать ее мы будем так:

<переменная>:=<выражение>

Значок «:=» читается «присвоить». Например:

$Z:=X+Y$

Комьютер сначала вычисляет выражение, затем результат присваивает переменной, стоящей слева от знака «:=».

Если до выполнения этой команды содержимое ячеек, соответствующих переменным X, Y, Z, было таким:

X	2	Y	5	Z	—

то после выполнения команды оно станет следующим:

X	2	Y	5	Z	7

Прочерк в ячейке Z обозначает, что начальное число в ней может быть любым. Оно не имеет значения для результата данной команды.

Если слева от знака присваивания стоит числовая переменная, а справа — выражение, определяющее порядок вычисления числовой величины, то такую команду называют **арифметической командой присваивания**, а выражение — **арифметическим выражением.**

В частном случае арифметическое выражение в правой части оператора присваивания может быть представлено одной переменной или одной константой. Например:

$X:=5$
$Y:=X$

Команда ввода

Значения переменных, являющихся исходными данными решаемой задачи, как правило, задаются **вводом.**

Команда ввода в описаниях алгоритмов выглядит так:

ввод <список переменных>.

Например:

ввод A, B, C

Пользователю удобно, если ввод данных организован в режиме диалога, когда по команде ввода компьютер прерывает выполнение программы и ждет действий пользователя. Пользователь должен набрать на клавиатуре вводимые значения переменных и нажать клавишу <ВВОД>. Введенные значения присвоятся соответствующим переменным из списка ввода, и выполнение программы продолжится.

Вот схема выполнения приведенной выше команды.

1. Память до выполнения команды:

A	—	B	—	C	—

2. Процессор компьютера получил команду *ввод* A, B, C, прервал свою работу и ждет действий пользователя.
3. Пользователь набирает на клавиатуре:

1 3 5

и нажимает клавишу <ВВОД> (<Enter>).

4. Память после выполнения команды:

A	1	B	3	C	5

5. Процессор переходит к выполнению следующей команды программы.

При выполнении пункта 3 вводимые числа должны быть отделены друг от друга какими-нибудь разделителями. Обычно это пробелы.

Из сказанного выше можно сделать вывод:

Переменные величины получают конкретные значения в результате выполнения команды присваивания или команды ввода.

Если переменной величине не присвоено никакого значения (или не введено), то она является неопределенной. Иначе говоря, ничего нельзя сказать о том, какое значение имеет эта переменная.

Команда вывода

Результаты решения задачи сообщаются компьютером пользователю путем выполнения **команды вывода**.

Команда вывода в алгоритмах записывается так:

вывод <список вывода>

Например:

вывод $X1$, $X2$

По этой команде значения переменных $X1$ и $X2$ будут вынесены на устройство вывода (чаще всего это экран).

О других командах, применяемых в алгоритмах работы с величинами, вы узнаете позже.

Коротко о главном

Любой алгоритм работы с величинами может быть составлен из следующих команд: присваивание; ввод; вывод; обращение к вспомогательному алгоритму; цикл; ветвление.

Программа для компьютера — это алгоритм, записанный на языке программирования.

Язык программирования — это фиксированная система обозначений для описания алгоритмов и структур данных.

Всякая обрабатываемая программой величина занимает определенное поле в памяти компьютера. Значение величины — это информация, хранимая в этом поле.

Переменная величина получает значение в результате выполнения команды присваивания или команды ввода.

Формат команды присваивания:

<переменная>:=<выражение>

Сначала вычисляется выражение, затем полученное значение присваивается переменной.

Ввод — это занесение данных с внешних устройств в оперативную память компьютера. Исходные данные для решения задачи обычно задаются вводом.

Результаты решения задачи выносятся на устройства вывода (монитор, принтер) по команде вывода.

Вопросы и задания

1. Что такое величина? Чем отличаются переменные и постоянные величины?
2. Чем определяется значение величины?
3. Какие существуют основные типы величин в программировании?
4. Как записывается команда присваивания?
5. Что такое ввод? Как записывается команда ввода?

6. Что такое вывод? Как записывается команда вывода?

7. В схематическом виде (как это сделано в параграфе) отразите изменения значений в ячейках, соответствующих переменным A и B, в ходе последовательного выполнения команд присваивания:

1)	2)	3)
$A:=1$	$A:=1$	$A:=1$
$B:=2$	$B:=2$	$B:=2$
$A:=A+B$	$C:=A$	$A:=A+B$
$B:=2*A$	$A:=B$	$B:=A-B$
	$B:=C$	$A:=A-B$

8. Вместо многоточия впишите в алгоритм несколько команд присваивания, в результате чего должен получиться алгоритм возведения в четвертую степень введенного числа (дополнительные переменные не использовать):

ввод A . . . вывод A

ЕК ЦОР: часть 2, глава 6, § 33. ЦОР № 2, 7.

§ 10 Линейные вычислительные алгоритмы

Основные темы параграфа:

- *присваивание*; *свойства присваивания*;
- *обмен значениями двух переменных*;
- *описание линейного вычислительного алгоритма.*

Присваивание. Свойства присваивания

Поскольку присваивание является важнейшей операцией в алгоритмах, работающих с величинами, поговорим о ней более подробно.

Переменная величина получает значение в результате присваивания.

Присваивание производится компьютером при выполнении одной из двух команд из представленной выше системы команд: команды присваивания или команды ввода.

Рассмотрим последовательность выполнения четырех команд присваивания, в которых участвуют две переменные: a и b. В приведенной ниже таблице против каждой команды указываются значения переменных, которые устанавливаются после ее выполнения. Такая таблица называется **трассировочной таблицей**, а процесс ее заполнения называется **трассировкой** алгоритма.

Команда	***a***	***b***
$a:=1$	1	—
$b:=2 * a$	1	2
$a:=b$	2	2
$b:=a + b$	2	4

Прочерк в таблице означает неопределенное значение переменной. Конечные значения, которые получают переменные a и b, соответственно равны 2 и 4.

Этот пример иллюстрирует три основных свойства присваивания. Вот эти свойства:

1) пока переменной не присвоено значение, она остается неопределенной;
2) значение, присвоенное переменной, сохраняется вплоть до выполнения следующего присваивания этой переменной нового значения;
3) новое значение, присвоенное переменной, заменяет ее предыдущее значение.

Обмен значениями двух переменных

Рассмотрим еще один очень полезный алгоритм, с которым при программировании часто приходится встречаться. Даны две переменные величины: X и Y. Требуется произвести между ними обмен значениями. Например, если первоначально было: $X = 1$; $Y = 2$, то после обмена должно стать: $X = 2$, $Y = 1$.

Хорошим аналогом для решения такой задачи является следующая: даны два стакана, в первом — молоко, во втором — вода; требуется произвести обмен их содержимым. Всякому

ясно, что в этом случае нужен дополнительный, третий, пустой стакан. Последовательность действий будет следующей:

1) перелить из 1-го стакана в 3-й;
2) перелить из 2-го стакана в 1-й;
3) перелить из 3-го стакана во 2-й.

Цель достигнута!

По аналогии для обмена значениями двух переменных нужна третья дополнительная переменная. Назовем ее Z. Тогда задача решается последовательным выполнением трех операторов присваивания (пусть начальные значения 1 и 2 для переменных X и Y задаются вводом):

Команда	X	Y	Z
ввод X, Y	1	2	–
Z:=X	1	2	1
X:=Y	2	2	1
Y:=Z	2	1	1
вывод X, Y	**2**	**1**	1

Действительно, в итоге переменные X и Y поменялись значениями. На экран будут выведены значения X и Y: 2, 1. В трассировочной таблице выводимые значения выделены жирным шрифтом.

Аналогия со стаканами не совсем точна в том смысле, что при переливании из одного стакана в другой первый становится пустым. В результате же присваивания (X:=Y) переменная, стоящая справа (Y), сохраняет свое значение.

Описание линейного вычислительного алгоритма

Алгоритмы, результатами выполнения которых являются числовые величины, будем называть вычислительными алгоритмами. Рассмотрим пример решения следующей математической задачи: даны две простые дроби; получить дробь, являющуюся результатом деления одной на другую.

В школьном учебнике математики правила деления обыкновенных дробей описаны так:

1. Числитель первой дроби умножить на знаменатель второй.

2. Знаменатель первой дроби умножить на числитель второй.

3. Записать дробь, числителем которой является результат выполнения пункта 1, а знаменателем — результат выполнения пункта 2.

В алгебраической форме это выглядит следующим образом:

$$\frac{a}{b} : \frac{c}{d} = \frac{a \cdot d}{b \cdot c} = \frac{m}{n}.$$

Теперь построим алгоритм деления дробей для компьютера. В этом алгоритме сохраним те же обозначения для переменных, которые использованы в записанной выше формуле. Исходными данными являются целочисленные переменные a, b, c, d. Результатом — также целые величины m и n.

Ниже алгоритм представлен в двух формах: в виде блок-схемы и на Алгоритмическом языке (АЯ).

Раньше прямоугольник в схемах алгоритмов управления мы называли блоком простой команды. Для вычислительных алгоритмов такой простой командой является команда присваивания. Прямоугольник *будем называть блоком присваивания, или вычислительным блоком.* В форме параллелограмма рисуется блок ввода/вывода. Полученный алгоритм имеет линейную структуру (рис. 2.1).

```
алг  Деление дробей
цел  a, b, c, d, m, n
нач
    ввод a, b, c, d
    m:=a×d
    n:=b×c
    вывод m, n
кон
```

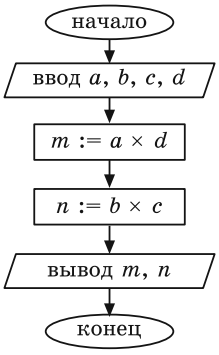


Рис. 2.1. Блок-схема алгоритма деления дробей

В алгоритме на АЯ строка, стоящая после заголовка алгоритма, называется **описанием переменных**. Служебное слово **цел** означает целый тип. Величины этого типа могут иметь только целочисленные значения.

Описание переменных имеет вид:

<тип переменных> <список переменных>

Список переменных включает все переменные величины данного типа, обрабатываемые в алгоритме.

В блок-схемах типы переменных не указываются, но подразумеваются. Запись алгоритма на АЯ ближе по форме к языкам программирования, чем блок-схемы.

Коротко о главном

Основные свойства присваивания:

- значение переменной не определено, если ей не присвоено никакого значения;
- новое значение, присваиваемое переменной, заменяет ее старое значение;
- присвоенное переменной значение сохраняется в ней вплоть до нового присваивания.

Обмен значениями двух переменных можно производить через третью дополнительную переменную.

Трассировочная таблица используется для «ручного» исполнения алгоритма с целью его проверки.

В алгоритмах на АЯ указываются типы всех переменных. Такое указание называется описанием переменных.

Числовые величины, принимающие только целочисленные значения, описываются с помощью служебного слова **цел** (целый).

Вопросы и задания

1. Из каких команд составляется линейный вычислительный алгоритм?
2. Что такое трассировка? Как она производится?
3. В каком случае значение переменной считается неопределенным?
4. Что происходит с предыдущим значением переменной после присваивания ей нового значения?
5. Как вы думаете, можно ли использовать в выражении оператора присваивания неопределенную переменную? К каким последствиям это может привести?
6. Напишите на АЯ алгоритм сложения двух простых дробей (без сокращения дроби).

7. Напишите на АЯ алгоритм вычисления y по формуле

$$y = (1 - x^2 + 5x^4)^2,$$

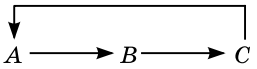
где x — заданное целое число. Учтите следующие ограничения: 1) в арифметических выражениях можно использовать только операции сложения, вычитания и умножения; 2) каждое выражение может содержать только одну арифметическую операцию. Выполните трассировку алгоритма при $x = 2$.

8. Пользуясь ограничениями предыдущей задачи, напишите наиболее короткие алгоритмы вычисления выражений:

$$y = x^8; \quad y = x^{10}; \quad y = x^{15}; \quad y = x^{19}.$$

Постарайтесь использовать минимальное количество дополнительных переменных. Выполните трассировку алгоритмов.

9. Запишите алгоритм циклического обмена значениями трех переменных A, B, C. Схема циклического обмена:



Например, если до обмена было: $A = 1$, $B = 2$, $C = 3$, то после обмена должно стать: $A = 3$, $B = 1$, $C = 2$. Выполните трассировку.

www ЕК ЦОР: часть 2, глава 6, § 34. ЦОР № 9, 10.

§ 11
Знакомство с языком Паскаль

Основные темы параграфа:

- *возникновение и назначение Паскаля*;
- *структура программы на Паскале*;
- *операторы ввода, вывода, присваивания*;
- *правила записи арифметических выражений*;
- *пунктуация Паскаля.*

Возникновение и назначение Паскаля

После того как построен алгоритм решения задачи, составляется программа на определенном языке программирования.

Среди современных языков программирования одним из самых популярных является язык **Паскаль**. Этот язык разработан в 1971 году и назван в честь Блеза Паскаля — французского ученого, изобретателя механической вычислительной

машины. Автор языка Паскаль — швейцарский профессор Никлаус Вирт.

Паскаль — это универсальный язык программирования, позволяющий решать самые разнообразные задачи обработки информации.

Команду алгоритма, записанную на языке программирования, принято называть **оператором**.

Программа на Паскале близка по своему виду к описанию алгоритма на АЯ. Сравните алгоритм решения уже знакомой вам задачи деления простых дробей с соответствующей программой на Паскале:

алг Деление дробей	`Program Division;`
цел *a*, *b*, *c*, *d*, *m*, *n*	`var a, b, c, d, m, n: integer;`
нач	`begin`
ввод *a*, *b*, *c*, *d*	`readln(a, b, c, d);  {Ввод}`
m:= *a*×*d*	`m:= a*d;  {Числитель}`
n:= *b*×*c*	`n:= b*c;  {Знаменатель}`
вывод *m*, *n*	`write(m, n)  {Вывод}`
кон	`end.`

Структура программы на Паскале

Даже не заглядывая в учебник по Паскалю, в этой программе можно все понять (особенно помогает знание английского языка).

Заголовок программы начинается со слова **Program** (программа), за которым следует произвольное имя, придуманное программистом:

Program <имя программы>;

Раздел описания переменных начинается со слова **Var** (variables — переменные), за которым идет список имен переменных через запятую. Тип указывается после двоеточия. В стандарте языка Паскаль существуют два типа числовых величин: **вещественный** и **целый**. Слово `integer` обозначает целый тип (является идентификатором целого типа). Вещественный тип обозначается словом `real`. Например, раздел описания переменных может быть таким:

```
var a, b: integer; c, d: real;
```

Идентификаторы переменных составляются из латинских букв и цифр; первым символом обязательно должна быть буква.

Раздел операторов — основная часть программы. Начало и конец раздела операторов программы отмечаются служебными словами **begin** (начало) и **end** (конец). В самом конце программы ставится точка:

```
begin
      <операторы>
end.
```

Операторы ввода, вывода, присваивания

Ввод исходных данных с клавиатуры происходит по оператору `read` (read — читать) или `readln` (read line — читать строку):

```
read(<список переменных>)
или  readln(<список переменных>)
```

При выполнении команды ввода компьютер ожидает действий пользователя. Пользователь набирает на клавиатуре значения переменных в том порядке, в каком переменные указаны в списке, отделяя их друг от друга пробелами. Одновременно с набором данных на клавиатуре они появляются на экране. В конце нажимается клавиша <ВВОД> (<Enter>). Разница в выполнении операторов `readln` и `read` состоит в том, что после выполнения ввода по оператору `readln` экранный курсор перемещается в начало новой строки, а по оператору `read` этого не происходит.

Вывод результатов происходит по оператору `write` (write — писать) или `writeln` (write line — писать в строку):

```
write(<список вывода>)
или  writeln(<список вывода>)
```

Результаты выводятся на экран компьютера в порядке их перечисления в списке. Элементами списка вывода могут быть константы, переменные, выражения.

Разница в выполнении операторов `writeln` и `write` состоит в том, что после выполнения вывода по оператору `writeln` экранный курсор перемещается в начало новой строки, а по оператору `write` этого не происходит.

Арифметический оператор присваивания на Паскале имеет следующий формат:

```
<числовая переменная>:=<арифметическое выражение>
```

Арифметическое выражение может содержать числовые константы и переменные, знаки арифметических операций, круг-

лые скобки. Кроме того, в арифметических выражениях могут присутствовать функции.

Знаки основных арифметических операций записываются так:

\+ сложение,
– вычитание,
* умножение,
/ деление.

Правила записи арифметических выражений

Запись арифметических выражений на Паскале похожа на обычную математическую запись. В отличие от математики, где часто пропускается знак умножения (например, пишут $2A$), в Паскале этот знак пишется обязательно: `2*A`. Например, математическое выражение

$$A^2 + B^2 - 12C$$

на Паскале записывается так:

```
A*A + B*B - 12*C
```

Это же выражение можно записать иначе:

```
SQR(A) + SQR(B) - 12*C
```

Здесь использована функция возведения в квадрат — SQR. Аргументы функций всегда пишутся в круглых скобках.

Последовательность выполнения операций определяется по их **приоритетам** (старшинству). К старшим операциям относятся умножение (*) и деление (/). Операции сложения и вычитания — младшие. В первую очередь выполняются старшие операции. Несколько операций одинакового старшинства, записанные подряд, выполняются в порядке их записи слева направо. Приведенное выше арифметическое выражение будет вычисляться в следующем порядке (порядок вычислений указан цифрами сверху):

```
    1     4     2     5     3
A   *   A   +   B   *   B   -   12   *   C
```

Круглые скобки в арифметических выражениях влияют на порядок выполнения операций. Как и в математике, в первую очередь выполняются операции в скобках. Если имеется несколько пар вложенных скобок, то сначала выполняются операции в самых внутренних скобках. Например:

```
    6       1       3     2       4     5
A + ((C - D) / (2 + K) - 1) * B
```

Пунктуация Паскаля

Необходимо строгое соблюдение правописания (синтаксиса) программы. В частности, в Паскале однозначно определено назначение знаков пунктуации.

Точка с запятой (;) ставится в конце заголовка программы, в конце раздела описания переменных, является разделителем описания переменных в разделе переменных и разделителем операторов. Перед словом **end** точку с запятой можно не ставить.

Запятая (,) является разделителем элементов во всевозможных списках: списке переменных в разделе описания, списках вводимых и выводимых величин.

Текст программы заканчивается точкой.

Строгий синтаксис в языке программирования необходим потому, что *компьютер является формальным исполнителем программы*. Если, допустим, разделителем в списке переменных должна быть запятая, то любой другой знак будет восприниматься как ошибка. Если точка с запятой является разделителем операторов, то в качестве оператора компьютер воспринимает всю часть текста программы от одной точки с запятой до другой. Если программист забыл поставить «;» между какими-то двумя операторами, то компьютер будет принимать их за один с неизбежной ошибкой.

В программу на Паскале можно вставлять комментарии. Комментарий — это пояснение к программе, которое записывается в фигурных скобках. В комментариях можно использовать русские буквы. На исполнение программы комментарий никак не влияет.

Заметим, что в Паскале нет различия между строчными и прописными буквами. Например, для Паскаля тождественны следующие варианты записи: **begin**, **Begin**, **BEGIN**, **BeGiN**. Использование строчных или прописных букв — дело вкуса программиста.

Коротко о главном

Паскаль — универсальный язык программирования.

Программа на Паскале состоит из заголовка, описаний и операторов.

Заголовок программы:

Program <имя программы>;

Описание переменных:

```
var <список однотипных переменных>: <тип>; ...
```

Раздел операторов:

```
begin
        <операторы>
end.
```

Операторы ввода данных с клавиатуры:

```
read(<список ввода>),  readln(<список ввода>)
```

Операторы вывода на экран:

```
write(<список вывода>), writeln(<список вывода>)
```

Арифметический оператор присваивания:

```
<переменная>:=<арифметическое выражение>
```

Арифметическое выражение может содержать любое количество арифметических операций и функций.

Последовательность выполнения операций определяется расстановкой скобок и старшинством операций (приоритетами). Старшие операции: *, /; младшие операции: +, – .

Точка с запятой ставится в конце заголовка программы, в конце раздела описания переменных, является разделителем переменных в разделе переменных и разделителем операторов. Текст программы заканчивается точкой.

Вопросы и задания

1. Когда появился язык Паскаль и кто его автор?
2. Как записывается заголовок программы на Паскале?
3. Как записывается раздел описания переменных?
4. С какими типами числовых величин работает Паскаль?
5. Как записываются операторы ввода и вывода в Паскале?
6. Что такое оператор присваивания?
7. Как записываются арифметические выражения?
8. По каким правилам определяется порядок выполнения операций в арифметическом выражении?
9. Какая задача решается по следующей программе?

```
Program  Test;
var  A,  B,  C:  integer;
begin
     readln(A,B);
     C:=(A+B)*(B-A);
     writeln(C)
end.
```