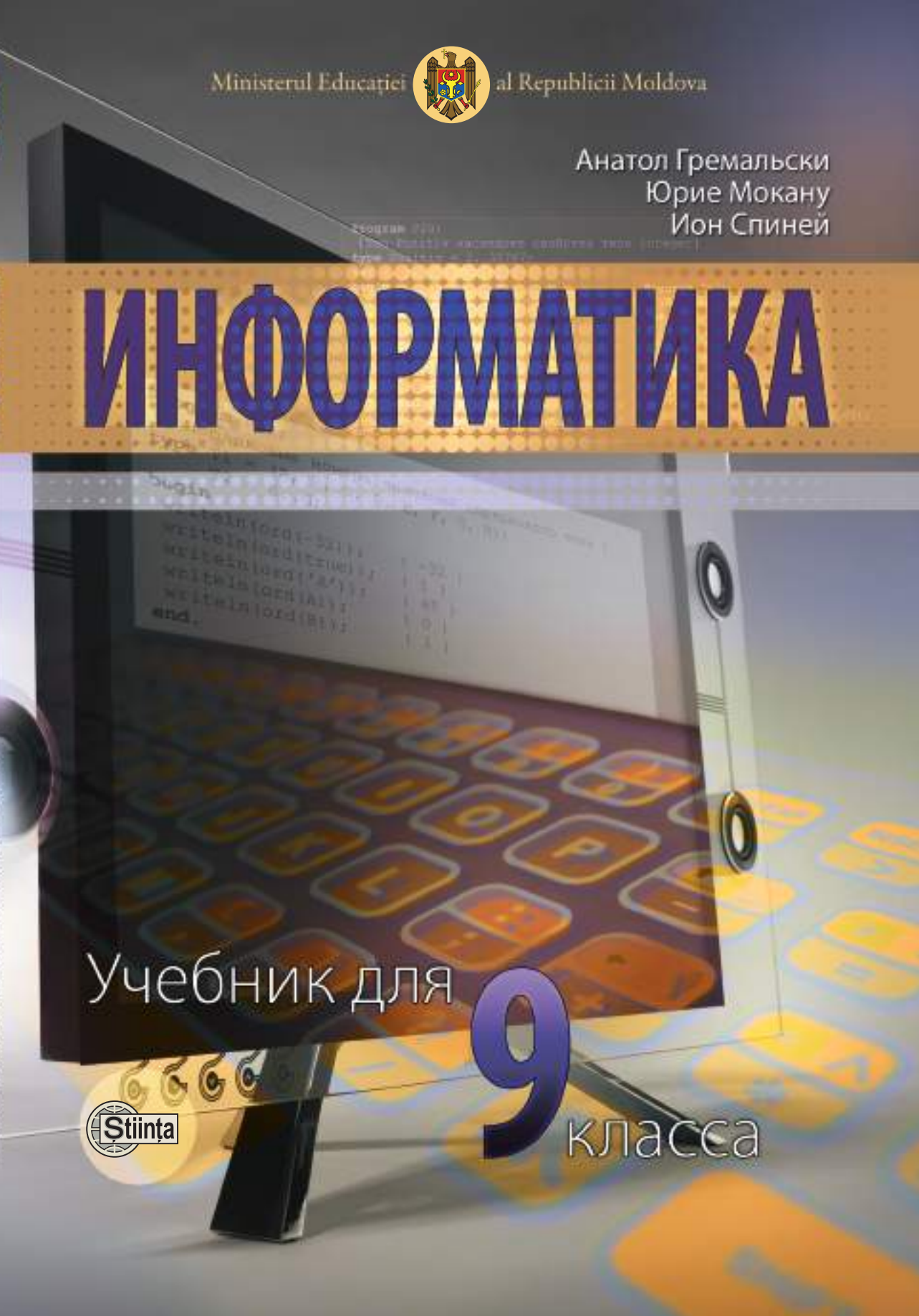

Ministerul Educației al Republicii Moldova
Анатол Гремальски
Юрие Мокану
Ион Спиней
ИНФОРМАТИКА
Учебник для 9 класса
Știința

Ministerul Educației al Republicii Moldova

Анатол Гремальски
Юрие Мокану
Ион Спиней

ИНФОРМАТИКА

Учебник для 9 класса

Știința, 2016

CZU 004(075.3)
Г 79

Elaborat conform Curriculumului disciplinar în vigoare și aprobat prin Ordinul ministrului educației (nr. 321 din 28 aprilie 2016). Editat din sursele financiare ale *Fondului special pentru manuale*.

Comisia de evaluare: *Ecaterina Adam*, prof. școlar (gr. did. I), Liceul de Creativitate și Inventică „Prometeu", Chișinău; *Mariana Ciobanu*, prof. școlar (gr. did. superior), Liceul Teoretic Român-Francez „Gheorghe Asachi", Chișinău; *Gheorghe Chistruga*, prof. școlar (gr. did. superior), Liceul Teoretic „Mihai Eminescu", Drochia; *Svetlana Golubev-Brînza*, prof. școlar (gr. did. superior), Liceul Teoretic „Nicolae Milescu-Spătaru", Chișinău; *Andrei Sacara*, prof. școlar (gr. did. superior), Liceul Teoretic „Miguel de Cervantes Saavedra", Chișinău

Recenzenți: *Tatiana Baciu*, doctor în psihologie, conferențiar universitar, Universitatea Pedagogică de Stat „Ion Creangă"; *Tatiana Cartaleanu*, doctor în filologie, conferențiar universitar, Universitatea Pedagogică de Stat „Ion Creangă"; *Alexei Colîbneac*, profesor universitar, Academia de Muzică, Teatru și Arte Plastice, maestru în arte

Redactor: *Valentina Rîbalchina*
Redactor tehnic: *Nina Duduciuc*
Machetare computerizată: *Anatol Andrițchi*
Copertă: *Vitalie Ichim*

Întreprinderea Editorial-Poligrafică Știința,
str. Academiei, nr. 3; MD-2028, Chișinău, Republica Moldova;
tel.: (+373 22) 73-96-16; fax: (+373 22) 73-96-27;
e-mail: prini@stiinta.asm.md; prini_stiinta@yahoo.com;
www.editurastiinta.md

DIFUZARE
ÎM Societatea de Distribuție a Cărții *PRO-NOI*,
str. Alba-Iulia, nr. 75; MD-2051, Chișinău;
tel.: (+373 22) 51-68-17, 71-96-74; fax: (+373 22) 58-02-68;
e-mail: info@pronoi.md; www.pronoi.md

Descrierea CIP a Camerei Naționale a Cărții

Гремальски, Анатол
Информатика: Учеб. для 9 класса/Анатол Гремальски, Юрие Мокану, Ион Спиней; comisia de evaluare: Ecaterina Adam [et al.]; trad. din lb. rom. Veronica Musteață; Min. Educației al Rep. Moldova. – Ch.: Î.E.P. *Știința*, 2016 (Tipogr. „BALACRON" SRL). – 144 p.

ISBN 978-9975-85-014-8

004(075.3)

ISBN 978-9975-85-014-8

Содержание

Дорогие друзья,

Вам уже известно, что информатика является областью науки, изучающей способы хранения, передачи и обработки информации с помощью компьютеров.

В 7–8 классах вы изучили основные понятия информатики – данные, информация, компьютер, исполнитель, алгоритм – и приобрели практические навыки работы на компьютере. Вы научились создавать и обрабатывать самые разнообразные документы с помощью текстового редактора, систематизировать и обрабатывать числовые и текстовые данные с помощью табличного процессора. Вы начились разрабатывать и отлаживать программы для управления исполнителями. В результате обучения вы убедились, что работой современных компьютеров можно управлять с помощью программ.

Данный учебника позволит усвоить знания, необходимые для развития алгоритмического мышления и формирования информационной культуры. Результатом обучения будет дальнейшее развитие способностей каждого ученика разрабатывать алгоритмы для решения возникающих в повседневной жизни задач с помощью компьютеров.

Известно, что алгоритмы очень точно описывают необходимые для обработки информации операции и порядок их следования. Обычно, в процессе начальной разработки алгоритма, пользователи используют для его описания разговорный язык, например, румынский, русский или английский. Затем, для более точного описания предполагаемых действий, могут быть использованы, например, блок-схемы. Но, для того, чтобы алгоритм стал понятен компьютеру, он должен быть написан на одном из языков программирования.

В данном учебнике для дидактических целей используется язык программирования ПАСКАЛЬ. Названный так в честь великого математика и философа Блеза Паскаля, этот язык получил мировое распространение благодаря тому, что он наиболее полно подходит для преподавания и изучения информатики. Будучи инструментом, предназначенным для развития алгоритмического мышления, язык ПАСКАЛЬ включает в себе все базовые концепции современных информационных систем: переменные, константы, типы данных, простые и составные операторы.

За каждым блоком теоретического материала, приведенного в учебнике, следуют примеры, упражнения и контрольные вопросы. Предполагается, что ученики введут и выполнят на компьютерах все приведенные в учебнике программы, ответят на поставленные вопросы, самостоятельно напишут на языке ПАСКАЛЬ и отладят программы, необходимые для решения предложенных задач. Все вошедшие в учебник программы были отлажены и протестированы в интегрированной среде программирования Turbo PASCAL 7.0. Очевидно, ученики и учителя могут использовать и другие программные продукты, предназначенные для процесса обучения с помощью компьютеров.

Будучи неотделимой частью современной культуры, информатика играет решающую роль в развитии человечества, открывает многообещающие перспективы каждому из нас, живущих и работающих в обществе, называемом, очень точно, информационным. Реализация указанных перспектив во многом зависит от знания фундаментальных принципов языков программирования и способностей их применения на практике.

Успехов вам!

Авторы

Повторение:

АЛГОРИТМЫ, ПРОГРАММЫ И ИСПОЛНИТЕЛИ

Известно, что *алгоритм* представляет собой конечную совокупность правил и предписаний, выполнение которых обеспечивает решение поставленной задачи. Процесс разработки алгоритмов называется алгоритмизацией.

В информатике понятие алгоритма неразрывно связано с понятием исполнителя. *Исполнитель* представляет собой объект, который может выполнять (исполнять) определенные команды. Множество таких команд образует систему команд исполнителя.

В 8-м классе мы изучили исполнителей Кенгуренок и Муравей, разработанных в учебных целях для школ нашей страны. Напомним, что исполнитель Кенгуренок может исполнять команды ШАГ, ПОВОРОТ, ПРЫЖОК, а исполнитель Муравей – команды ВВЕРХ, ВНИЗ, ВПРАВО, ВЛЕВО.

Существует два режима управления исполнителями: ручной и автоматический. *Режим ручного управления* предполагает что ввод каждой команды приводит к ее немедленному выполнению исполнителем. *Режим автоматического управления* предполагает выполнение последовательности команд без вмешательства пользователя. Перед использованием автоматического режима, алгоритм необходимо сначала написать и ввести его в память центра управления исполнителем.

Последовательность команд, предназначенная для автоматического управления исполнителем, называется *программой.* Очевидно, программа представляет собой алгоритм, записанный на языке исполнителя. Процесс разработки программ называется *программированием.*

В 8-м классе мы разработали большое число программ для управления исполнителями Кенгуренок и Муравей: рисование квадратов, орнаментов и спиралей, размещение символов в заданном порядке и др. Созданные нами программы были написаны на языке программирования и содержали простые операторы ШАГ, ПОВОРОТ, ПРЫЖОК, ВВЕРХ, ВНИЗ, ВПРАВО, ВЛЕВО, вызовы процедур (подпрограмм) и составные операторы ПОВТОРИ, ПОКА, ЕСЛИ.

Будучи разработанными в учебных целях, исполнители Кенгуренок и Муравей не могут обеспечить более сложную обработку информации. Очевидно, для решения практических задач, возникающих в реальной жизни, нужны более мощные исполнители, а именно, современные компьютеры.

Вообще говоря, компьютер является исполнителем, который выполняет в автоматическом режиме загруженные в его внутреннюю память программы. Напомним, что любая программа, загруженная во внутренней памяти компьютера, представляет

собой последовательность двоичных слов, которая указывает компьтеру состав и порядок требуемых операций.

В качестве примера, ниже приведен фрагмент программы, написанной на машинном языке:

```
10010101 10000011 00110100 01000100
01010010 01011101 00010010 10010101
11010010 01001100 00101001 01110100
00010101 01010100 11111010 10100011
```

Так как разработка программ в двоичных кодах является трудоемкой и неэффективной работой, алгоритмы, предназначенные для решения задач с помощью компьютера, целесообразно писать на специальных языках, называемых языками высокого уровня. Наиболее распространенными языками высокого уровня являются АЛГОЛ, ФОРТРАН, БЭЙСИК, ПАСКАЛЬ, СИ, ЯВА и др. Все эти языки содержат средства для написания и вызова подпрограмм, операторы для программирования линейных, разветвляющихся и циклических алгоритмов. В школах нашей страны, в учебных целях, используется язык ПАСКАЛЬ, наиболее полно подходящий для преподавания и изучения информатики.

Для примера, представляем программу написанную на языке ПАСКАЛЬ, которая вычисляет корни уравнения первой степени:

```
Program Exemplu;
var a, b, x : real;
begin
  readln(a, b);
  if a<>0 then
    begin
       x:=-b/a;
       writeln('Уравнение имеет единственный корень');
       writeln(x);
    end;
  if   (a=0) and (b=0) then
      writeln('Уравнение имеет бесконечное
            множество корней');
  if (a=0) and (b<>0) then
     writeln('Уравнение не имеет смысла');
end.
```

Сравнив фрагмент программы, написанный на машинном языке, с программой `Exemplu` можно убедиться, что применение языка ПАСКАЛЬ значительно упрощает процесс разработки программ.

В общем случае, при использованием языков высокого уровня, решение задачи с помощью компьютера включает следующие этапы:

1) описание алгоритма в общих чертах с помощью языка общения между людьми, например, русского, румынского или английского;

2) если необходимо, более точное описание требуемых операций с помощью блок-схем;

3) написание алгоритма на одном из языков высокого уровня, например, на ПАСКАЛЕ;

4) перевод программы с языка высокого уровня на машинный язык, то есть, в последовательность двоичных слов;

5) обнаружение и исправление возможных ошибок и запуск программы на выполнение.

Перевод программ с языка высокого уровня на машинный язык называется *компиляцией* и осуществляется в автоматическом режиме с помощью специальных программ, называемых *компиляторами*.

Вопросы и упражнения

❶ Вспомните, по крайней мере, три алгоритма, которые вы изучали на уроках математики и информатики.
❷ Какую информацию должно содержать полное описание определенного исполнителя?
❸ Каковы основные средства, предназначенные для представления алгоритмов?
❹ Чем отличается программный режим управления от ручного?
❺ В чем состоит отличие алгоритма от программы? Обоснуйте ваш ответ.
❻ Назовите основные этапы решения задач с помощью компьютеров.
❼ Чем отличаются языки высокого уровня от машинных языков?

Глава 1

СЛОВАРЬ И СИНТАКСИС ЯЗЫКА ПАСКАЛЬ

1.1. Знакомство с языком ПАСКАЛЬ

Рассмотрим следующую программу:

```
1  Program P1;
2   { Сумма целых чисел x, y, z }
3  var x, y, z, s : integer;
4  begin
5    writeln('Введите целые числа x, y, z:');
6    readln(x, y, z);
7    s:=x+y+z;
8    writeln('Сумма введенных чисел:');
9    writeln(s);
10 end.
```

Числа *1, 2, 3, …, 10* в левой части страницы не являются частью программы на языке ПАСКАЛЬ. Они служат только для цифрового обозначения объяснения значения каждой строки.

Строка 1. Слово **program** является зарезервированным словом языка, а `P1` – словом, определяемым пользователем. Зарезервированные слова служат для составления программ, а слова, определяемые пользователем, – для имен переменных, констант, подпрограмм, программ и т. д.

Строка 2. Данная строка является пояснением, комментарием. Комментарий заключается в фигурные скобки: {, }. Комментарий не влияет на работу программы и предназначен исключительно для пользователя.

Строка 3. Зарезервированное слово **var** (*variable* – переменная) описывает переменные `x, y, z` и `s`, используемые в программе. Слово `integer` (целый) указывает тип соответствующих переменных. Таким образом, переменные `x, y, z` и `s` могут принимать только целые значения. Данная строка образует раздел описаний.

Строка 4. Зарезервированное слово **begin** (начало) означает начало выполняемой части программы – раздела операторов.

Строка 5. Вывод сообщения на стандартное устройство вывода, обычно на экран. Слово `writeln` (*write line* – запись и переход на новую строку) представляет собой обращение к стандартной процедуре, аргументом которой является текст сообщения:

```
Введите целые числа x, y, z:
```

Отметим, что апострофы не являются частью текста отображаемого на экране.

Строка 6. Считывание трех чисел со стандартного устройства ввода, как правило, – с клавиатуры. Числа вводятся в одной строке и отделяются друг от друга одним или несколькими пробелами. После ввода последнего числа нажимается клавиша <ENTER>. Числа считываются в переменные `x, y, z`. Слово `readln` (*read line* – считывание и переход на новую строку) представляет собой обращение к стандартной процедуре. Аргументами данной процедуры являются имена переменных, в которых запоминаются введенные целые числа.

Строка 7. Оператор присваивания. Переменной `s` присваивается значение `x + y + z`.

Строка 8. Вывод сообщения

```
Сумма введенных чисел:
```

на стандартное устройство вывода.

Строка 9. Вывод значения переменной `s` на стандартное устройство вывода.

Строка 10. Зарезервированное слово **end** означает конец раздела операторов, а точка – конец программы.

Таким образом, программа на языке ПАСКАЛЬ состоит из следующих частей:

- **заголовок**, в котором указывается имя программы (в нашем примере строка 1);
- **раздел описаний**, где описываются используемые в программе переменные, функции, процедуры и т. д. (в нашем примере строка 3);
- **раздел операторов**, содержащий команды, которые компьютер должен выполнить в заданном порядке (в нашем примере строки 4–10).

Для редактирования, компиляции и выполнения программ, написанных на языке ПАСКАЛЬ, были разработаны специальные приложения, называемые средами программирования. Как правило, в школьных компьютерных классах установлена среда программирования "Turbo PASCAL 7.0". Графический интерфейс данного приложения содержит ряд меню, обеспечивающих выполнение следующих команд:

- ввод и редактирование программ ПАСКАЛЬ;
- сохранение программ ПАСКАЛЬ в виде отдельных файлов;
- чтение, редактирование и сохранение текстовых файлов и файлов, содержащих программы на языке ПАСКАЛЬ;
- отладка программ ПАСКАЛЬ;
- компиляция и запуск на выполнение программ ПАСКАЛЬ.

Вопросы и упражнения

❶ Введите и запустите на выполнение программу `P1`.

❷ Из каких основных частей состоит программа на языке ПАСКАЛЬ?

❸ Введите и запустите на выполнение следующую программу:

```
Program P2;
  { Вывод предописанной константы MaxInt }
begin
```

```
  writeln('MaxInt=', MaxInt);
end.
```

❹ Укажите заголовок, раздел описаний и раздел операторов программы P2. Объясните назначение каждой строки данной программы.

❺ Измените программу P1 таким образом, чтобы она вычисляла сумму чисел x, y введенных с клавиатуры.

1.2. Метаязык БНФ

Любой язык программирования определяется через синтаксис и семантику. Известно, что *синтаксис* – это совокупность правил, которые задают структуру предложений, а *семантика* – это совокупность правил, определяющих смысл и значение соответствующих предложений. В случае языков программирования эквивалентом *предложения* является *программа*.

Ясно, что синтаксис любого языка программирования может быть описан с помощью одного из языков общения между людьми, например, русского, румынского, английского, французского и т.д. Однако такого рода описание будет объемным и может оказаться двусмысленным. Поэтому для лаконичного и точного описания синтаксиса языков программирования были разработаны специальные языки, называемые *метаязыками*. Самым распространенным метаязыком является метаязык БНФ – *формы Бэкуса-Наура.*

Метаязык БНФ использует следующие символы:

• **терминальные символы** – символы, из которых состоит программа на языке ПАСКАЛЬ;

• **нетерминальные символы** – символы, которые обозначают грамматические единицы (конструкции) языка.

Нетерминальные символы записываются между знаками “<” и “>”.

Например, цифры 0, 1, 2 …, 9, буквы A, B, C, …, Z являются терминальными символами, а <*Цифра*>, <*Буква*> являются нетерминальными символами.

Описание синтаксиса языка ПАСКАЛЬ состоит из совокупности металингвистических формул.

Под **металингвистической формулой** будем понимать конструкцию, состоящую из двух частей: левой и правой, разделенных символами: “::=”, что означает «является по определению». В левой части формулы находится нетерминальный символ.

В правой части металингвистической формулы с помощью символа “|”, означающего «или», описываются всевозможные варианты определения нетерминального символа.

Например, формула

<*Цифра*> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

определяет грамматическую единицу <*Цифра*> как один из терминальных символов: 0, 1, ..., 9.

Таким же образом интерпретируется металингвистическая формула:

<*Буква*> ::= a | b | c | d | e | f | g | h | i | j | k | l |m |
n | o | p | q | r | s | t | u | v | w | x | y | z

В правой части металингвистической формулы может быть указана последовательность из двух и более символов. Такая запись соответствует операции **конкатенации** (присоединения).

В частности, формула

<Ид> ::= <Буква><Цифра>

определяет грамматическую конструкцию *<Ид>* как букву, за которой следует цифра.

Пример:

1) `a9`

2) `a1`

3) `c3`

4) `x4`

5) `d0`

6) `e8`

В случае, если некоторая часть определения нетерминального символа может отсутствовать или повторяться произвольное число раз, оно заключается в фигурные скобки: {, }.

Например, формула

<Целое без знака> ::= <Цифра> {<Цифра>}

определяет нетерминальный символ *<Целое без знака>* как непустую последовательность цифр. Последовательности `0`, `0000`, `001`, `1900`, `35910` соответствуют данному определению, а последовательность `3a5910` – не соответствует.

Формула

<Идентификатор> ::= <Буква> {<Буква> | <Цифра>}

указывает, что идентификатор всегда начинается с буквы, после которой может следовать конечная последовательность из букв и цифр. Например, `a`, `a1`, `a1b`, `a23x`, `a14bxz`, `ab5` соответствуют данной формуле, а `2a` – не соответствует.

В случае, когда некоторая часть определения нетерминального символа может отсутствовать или присутствовать ровно один раз, оно заключаются в квадратные скобки: [,].

Например, формула

<Масштабный множитель> ::= [+ | –] <Целое без знака>

определяет масштабный множитель как целое число без знака, которому может предшествовать + или –. Например: `1`, `+1`, `-1`, `20`, `+20`, `-20`, `+003` соответствуют данной формуле, а `3-5` – не соответствуют.

Обратим внимание на то, что символы [,], {, } принадлежат метаязыку и их не следует путать с соответствующими символами, используемыми в программах на языке ПАСКАЛЬ.

Вопросы и упражнения

❶ Дайте объяснение терминам *синтаксис* и *семантика*.

❷ Для чего предназначен метаязык?

❸ Как задается синтаксис любого языка программирования с помощью метаязыка БНФ?

❹ Синтаксис некоторого простого языка программирования описан с помощью следующих металингвистических формул:

<*Цифра*> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
<*Число*> ::= <*Цифра*> {<*Цифра*>}
<*Знак*> ::= + | –
<*Арифметическое выражение*> ::= <*Число*> {<*Знак*><*Число*>}

Какие из приведенных ниже последовательностей соответствуют определению лексической единицы <*Число*>?

a)	`0`	*f)*	`0+0`	*k)*	`0000`
b)	`1`	*g)*	`11100+1`	*l)*	`0001`
c)	`11100`	*h)*	`11100–1`	*m)*	`–152`
d)	`00011`	*i)*	`931`	*n)*	`+351`
e)	`20013`	*j)*	`614`	*o)*	`412`

Какие из приведенных ниже последовательностей соответствуют определению лексической единицы <*Арифметическое выражение*>?

a)	`0+1`	*f)*	`–13`	*k)*	`21+00000`
b)	`1+0–3`	*g)*	`21+–16`	*l)*	`39+00001`
c)	`0+0+4`	*h)*	`–21–16`	*m)*	`00001–00001`
d)	`1+1–9`	*i)*	`68–13`	*n)*	`379–486`
e)	`6+6+21`	*j)*	`42+650`	*o)*	`31+12–51+861`

❺ Синтаксис простого языка общения компьютер–пользователь задается следующим образом:

<*Диск*> ::= `A: | B: | C: | D: | E:`
<*Список параметров*> ::= <*Диск*> {, <*Диск*>}
<*Название команды*> ::= `Считывание | Копирование | Форматирование`
<*Команда*> ::= <*Название команды*> <*Список параметров*>

Какие из приведенных ниже последовательностей соответствуют определению лексической единицы <*Команда*>?

a)	`Считывание`	*f)*	`Копирование A: B:`
b)	`Считывание A:`	*g)*	`Считывание D`
c)	`Копирование F:`	*h)*	`Форматирование D:, F:`
d)	`Копирование A:,`	*i)*	`Копирование E:, A:,`
e)	`Форматирование D:, E:`	*j)*	`Копирование F:, A:`

1.3. Синтаксические диаграммы

Синтаксические диаграммы более наглядно описывают синтаксис языков программирования. Диаграммы составляются в соответствии с формулами БНФ.

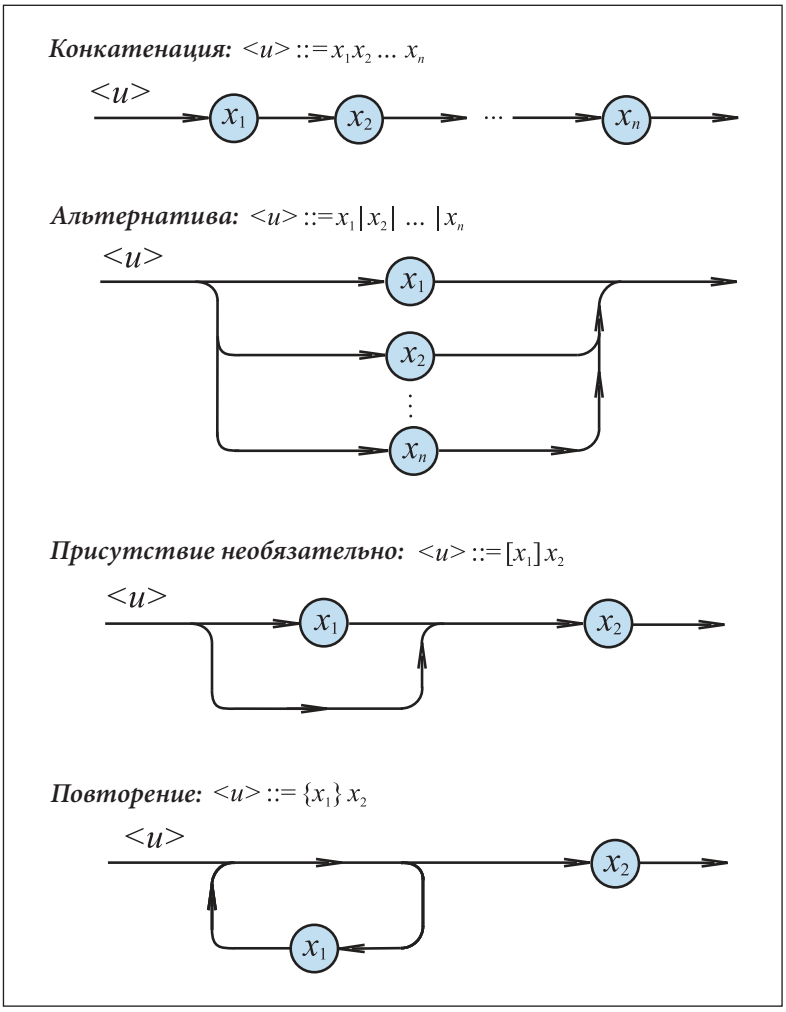


Рис. 1.1. Представление формул БНФ с помощью синтаксических диаграмм

Каждому терминальному символу на синтактических диаграммах должен соответствовать круг или овал, в который вписывается соответствующий символ. Нетерминальные символы заключаются в прямоугольник. Овалы и прямоугольники объединяются согласно синтактическим диаграммам, приведенным на *рис.1.1.*

На *рис. 1.2* показаны синтаксические диаграммы для грамматических единиц *<Целое без знака>*, *<Идентификатор>* и *<Масштабный множитель>*, определенных в предыдущем параграфе. Отметим, что каждому пути диаграммы соответствует синтаксически правильная последовательность терминальных символов.

Вопросы и упражнения

❶ Для чего предназначены синтаксические диаграммы?

❷ Как представляются терминальные и нетерминальные символы в синтаксических диаграммах?

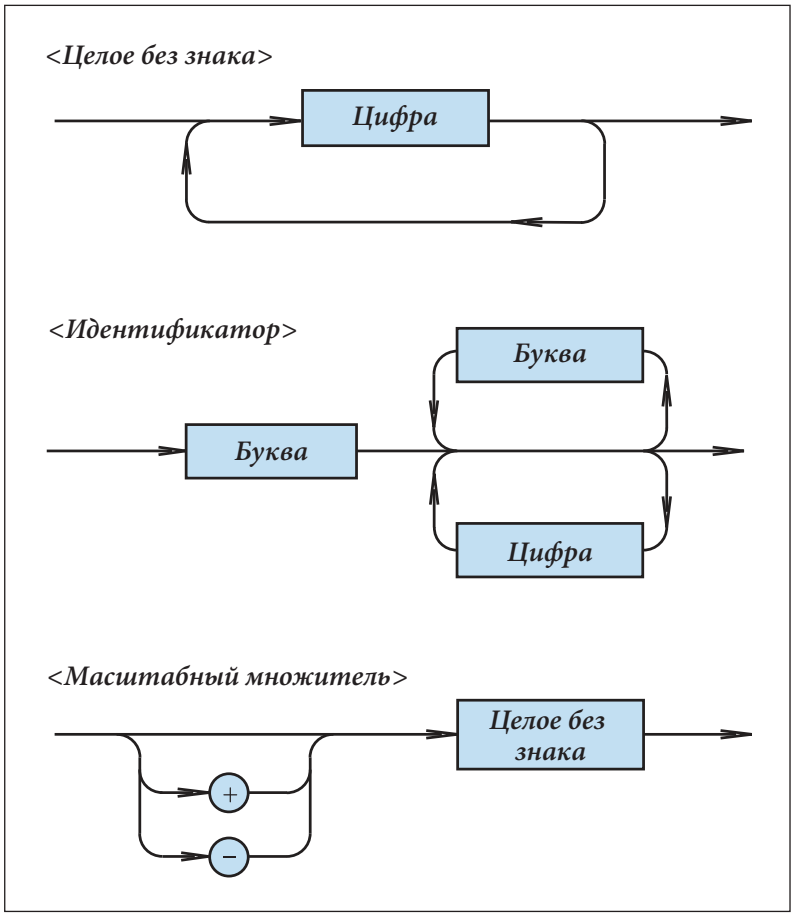


Рис. 1.2. Синтаксические диаграммы *<Целое без знака>*, *<Идентификатор>* и *<Масштабный множитель>*

❸ Как представляются формулы БНФ в виде синтаксических диаграмм?

❹ Представьте в виде синтаксических диаграмм:

<Цифра> ::= `0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`
<Число> ::= *<Цифра>* {*<Цифра>*}
<Знак> ::= `+ | –`
<Арифметическое выражение> ::= *<Число>*{*<Знак>**<Число>*}

❺ Представьте в виде синтаксических диаграмм:

<Диск> ::= `A:|B:|C:|D:|E:`
<Список параметров> ::= *<Диск>*{, *<Диск>*}
<Название команды> ::= `Считывание|Копирование|Форматирование`
<Команда> ::= *<Название команды>* *<Список параметров>*

❻ На *рис. 1.3* представлены синтаксические диаграммы, которые определяют грамматическую единицу *<Дробное число>*. Определите, какие из приведенных ниже примеров соответствуют данным диаграммам:

a)	`0.1`	*f)*	`.538`
b)	`+0.1`	*g)*	`721.386.`
c)	`-0.0`	*h)*	`-421`
d)	`9.000`	*i)*	`247.532`
e)	`-538.`	*j)*	`+109.000`

❼ Напишите формулы БНФ, соответствующие синтаксическим диаграммам *рис. 1.3.*

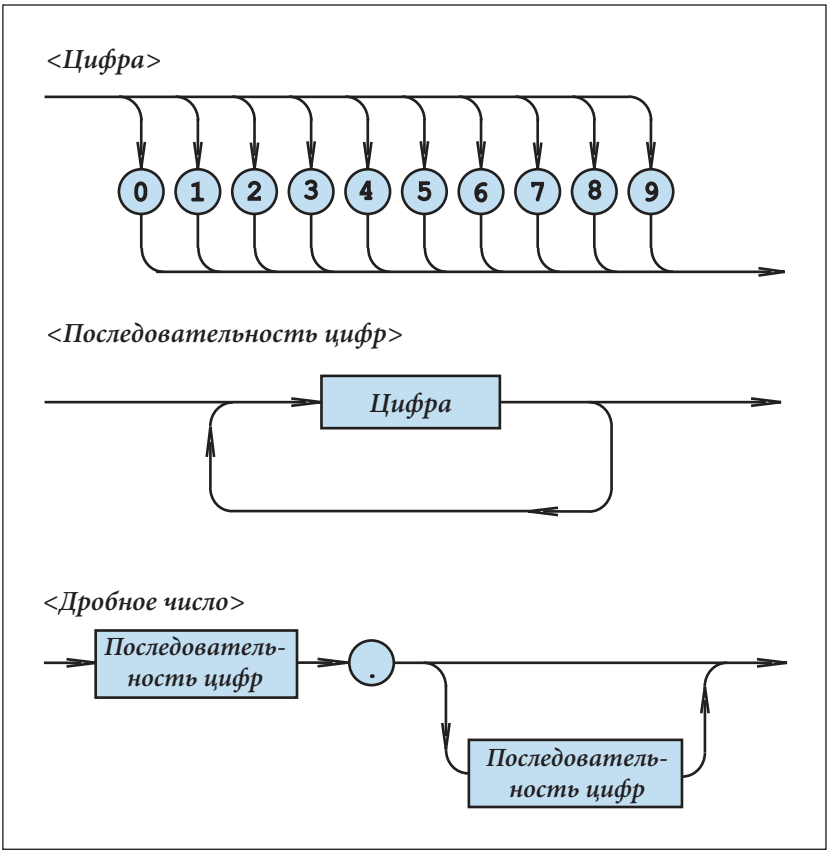


Рис. 1.3. Синтаксические диаграммы *<Цифра>*, *<Последовательность цифр>*, *<Дробное число>*

1.4. Алфавит языка ПАСКАЛЬ

Алфавит языка ПАСКАЛЬ имеет в своем составе следующие символы кода *ASCII* (*American Standard Code for Information Interchange*):

- десятичные цифры;
- прописные и строчные буквы латинского алфавита;

• знаки препинания;
• арифметические и логические операторы;
• управляющие символы (пробел, конец строки или возврат каретки и т. д.).

В некоторых конструкциях языка могут использоваться и буквы национальных алфавитов, например буквы ă, â, î, ş, ţ румынского алфавита, буквы русского алфавита и т. д.

1.5. Словарь языка ПАСКАЛЬ

Самые простые элементы языка ПАСКАЛЬ, составленные из символов и имеющие лингвистическое значение, называются *лексемами* или *лексическими единицами*. Они и составляют **словарь** языка ПАСКАЛЬ.

Существуют следующие **лексические единицы** языка ПАСКАЛЬ:
• специальные символы и ключевые слова;
• идентификаторы;
• числа;
• строки символов;
• метки;
• директивы.

1.5.1. Специальные символы и ключевые слова

Специальные символы языка ПАСКАЛЬ состоят из одного или двух символов:

+	плюс	<	меньше
–	минус	>	больше
*	звездочка	[	левая квадратная скобка
/	наклонная черта	]	правая квадратная скобка
=	равно	(	левая круглая скобка
,	запятая	)	правая круглая скобка
:	двоеточие	;	точка с запятой
.	точка	^	циркумфлекс
@	коммерческое at	$	доллар
{	левая фигурная скобка	<=	меньше либо равно
}	правая фигурная скобка	>=	больше либо равно
#	номер	:=	присваивание

`..`	многоточие	`<>`	не равно
`(*`	эквивалент фигурной скобки {	`(.`	эквивалент квадратной скобки [
`*)`	эквивалент фигурной скобки }	`.)`	эквивалент квадратной скобки]

Отметим, что специальные символы, составленные из двух знаков, например `<=` или `:=`, не допускают "вклинивания" в них разделителей.

Ключевые слова языка ПАСКАЛЬ состоят из двух или более букв:

`and`	и	**`nil`**	нуль
`array`	массив	**`not`**	нет
`begin`	начало	**`of`**	из
`case`	выбор	**`or`**	или
`const`	константа	**`packed`**	упакованный
`div`	целочисленное деление	**`procedure`**	процедура
`do`	выполнить	**`program`**	программа
`downto`	убывает до	**`record`**	запись
`else`	иначе	**`repeat`**	повторить
`end`	конец	**`set`**	множество
`file`	файл	**`then`**	тогда
`for`	для	**`to`**	до
`function`	функция	**`type`**	тип
`goto`	перейти на	**`until`**	до тех пор, пока
`if`	если	**`var`**	переменная
`in`	в	**`while`**	пока
`label`	метка	**`with`**	с
`mod`	остаток от деления		

Ключевые слова являются зарезервированными и не могут использоваться с целью, отличной от той, которая предназначена им по определению языка.

Лексические единицы, рассматриваемые в данном параграфе, могут быть определены следующими формулами БНФ:

<*Специальный символ*> ::= `+` | `-` | `*` | `/` | `=` | `<` | `>` | `]` | `[` | `,` | `(` | `)` | `:` | `;` | `^` | `.` | `@` | `{` | `}` | `$` | `#` | `<=` | `>=` | `<>` | `:=` | `..` | <*Ключевое слово*> | <*Эквивалентный символ*>

<*Эквивалентный символ*> ::= `(* | *) | (. | .)`

<*Ключевое слово*>::= **`and | array | begin | case | const | div | do | downto | else | end | file | for | function | goto | if | in | label | mod | nil | not | of | or | packed | procedure | program | record | repeat | set | then | to | type | until | var | while | with`**

Символы {, }, [и], используемые в формулах БНФ, являются одновременно и элементами языка ПАСКАЛЬ. Чтобы избежать двусмысленности, данные символы, как элементы словаря, могут быть представлены через эквивалентные символы `(*`, `*)` и соответственно `(.`, `.)`.

Вопросы и упражнения

❶ Запомните ключевые слова языка ПАСКАЛЬ.
❷ В чем разница между символами и специальными символами?
❸ Постройте синтаксические диаграммы для лексических единиц: <*Специальный символ*>, <*Эквивалентный символ*>, <*Ключевое слово*>.

1.5.2. Идентификаторы

Идентификаторы – это лексические единицы, которые выступают в качестве имен переменных, констант, функций, программ и т. д.

Любой идентификатор начинается с буквы, за которой может следовать любая комбинация из букв и цифр. Длина идентификаторов не ограничена, но только первые 63 символа являются значимыми.

Напомним формулы БНФ, определяющие лексическую единицу <*Идентификатор*>:

<*Цифра*> ::= `0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`

<*Буква*> ::= `a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z`

<*Идентификатор*> ::= <*Буква*> { <*Буква*> | <*Цифра*> }

Примеры идентификаторов:

1) `x`
2) `y`
3) `z`
4) `x1`
5) `y10`
6) `z01b`
7) `lista`
8) `listaelevilor`
9) `listatelefoanelor`
10) `registru`
11) `adresa`
12) `adresadomiciliu`
13) `clasa10`
14) `anul2011`

Прописные и строчные буквы в грамматических конструкциях языка ПАСКАЛЬ, за исключением строк символов, считаются эквивалентными. Таким образом, следующие идентификаторы являются эквивалентными:

1) `x и X`

2) `y и Y`

3) `z и Z`

4) `x1 и X1`

5) `y10 и Y10`

6) `z01b, Z01b, Z01B и z01B`

7) `lista, Lista, LIsta, ListA, LISTa`

Использование прописных и строчных букв позволяет записывать идентификаторы в форме, удобной для чтения, например:

1) `ListaElevilor`

2) `ListaTelefoanelor`

3) `AdresaDomiciliu`

4) `BugetulAnului2011`

Отметим, что в основных конструкциях языка ПАСКАЛЬ не используются буквы, присущие только русскому и румынскому алфавитам. Они заменяются на соответствующие буквы латинского алфавита:

1) `Suprafata`

2) `Numar`

3) `Gorod`

4) `Plosciadi`

4) `Patrat`

5) `SirDeCaractere`

6) `NumarIncercari`

6) `Nomer`

Вопросы и упражнения

❶ Нарисуйте синтаксические диаграммы для грамматических единиц <*Цифра*>, <*Буква*> и <*Идентификатор*>.

❷ Какие из приведенных ниже примеров соответствуют определению лексической единицы <*Идентификатор*>:

a) `x1`

b) `X1`

c) `1x`

d) `1X`

e) `xy`

f) `Suprafata`

g) `SUPRAFATA`

h) `rădăcina`

i) `radacina`

j) `R1`

k) `A1x`

l) `ListaA`

m) `Lista1`

n) `B-1`

o) `abc`

p) `Dreptunghi`

q) `iI`

r) `I1j`

s) `Luni` *t)* `Luna` *u)* `20.07.2011`

Для идентификаторов укажите соответствующий путь в синтаксической диаграмме *<Идентификатор>*.

❸ Найдите пары эквивалентных идентификаторов:

a) `x101`
b) `ya15`
c) `radacinaX1`
d) `radacinaX2`
e) `triunghi`
f) `cerc`
g) `sirdecaractere`
h) `registruclasa10`
i) `COTIDIAN`
j) `ZILE`
k) `CERCURI`
l) `SirDeCaractere`
m) `Triunghiuri`
n) `RegistruClasa10`
o) `zilE`
p) `X101`
q) `RegistruClasa10`
r) `radaciniX1X2`
s) `RADACINAX1`
t) `yA101`

❹ Для чего предназначены идентификаторы в программах языка ПАСКАЛЬ?

❺ Для нахождения корней x_1, x_2 квадратного уравнения $ax^2 + bx + c = 0$ сначала находится дискриминант d. Предложите несколько вариантов представления коэффициентов a, b, c, дискриминанта d и решений x_1, x_2 посредством идентификаторов.

1.5.3. Числа

Числа могут быть целыми или вещественными. Обычно используется десятичная система счисления. На *рис. 1.4* показаны синтаксические диаграммы для лексических единиц *<Целое число>* и *<Вещественное число>*.

Примеры целых чисел:

`23`	`–23`	`0023`	`+023`
`318`	`00318`	`+0318`	`–318`
`1996`	`+1996`	`–1996`	`0001996`
`–0023`	`–0318`	`+001996`	`-000199`

В случае вещественных чисел дробная часть отделяется от целой точкой. Перед десятичной точкой должна стоять, по крайней мере, одна десятичная цифра.

Примеры вещественных чисел:

`3.1415`	`+3.04`	`0.0001`	`283.19`
`–3.04`	`–0.0001`	`–256.19`	`28.17`
`+0.0001`	`6.28`	`+3.12421`	`63906.734`

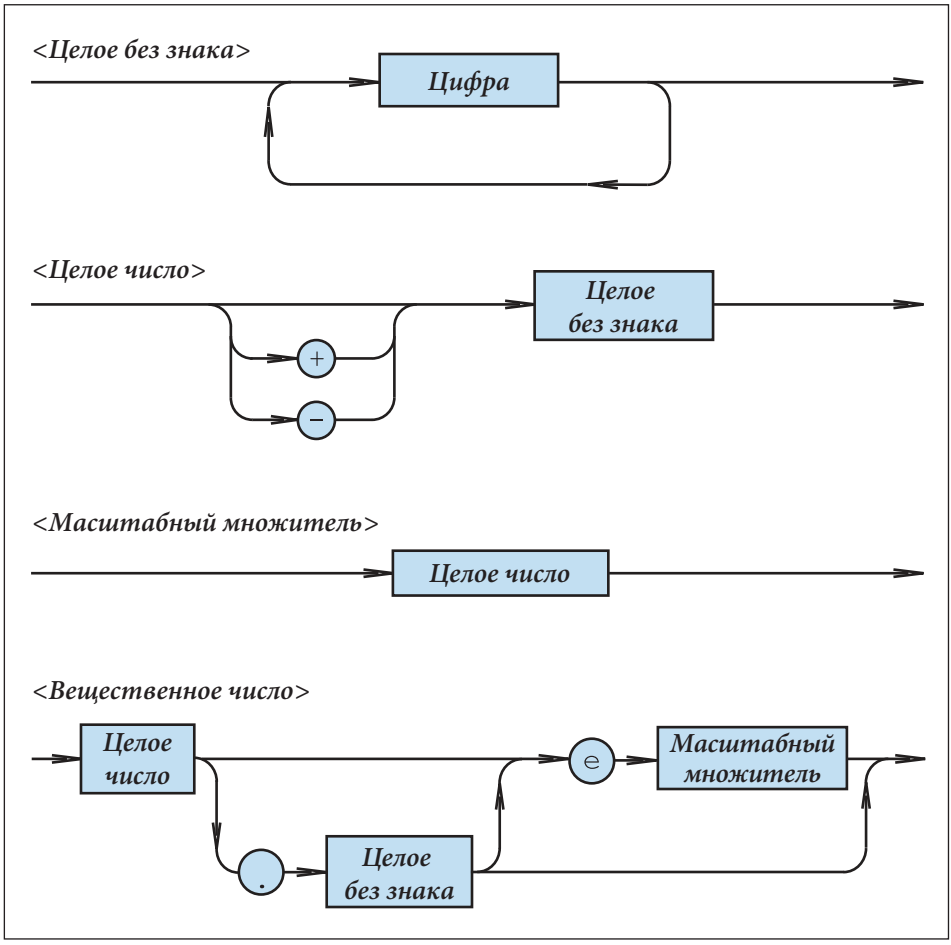


Рис. 1.4. Синтаксические диаграммы <*Целое число*> и <*Вещественное число*>

При записи вещественных чисел можно использовать **масштабный множитель**. Масштабный множитель представляет собой целое число, перед которым ставится буква `e` (или `E`), и означает, что число, за которым он следует, умножается на 10 в соответствующей степени.

Примеры:

	Обычная форма записи	*Запись на языке ПАСКАЛЬ*
1)	$8{,}12 \cdot 10^{-5}$	`8.12e-5`
2)	$749{,}512 \cdot 10^{8}$	`749.512e+8`
3)	$-0{,}0823 \cdot 10^{-12}$	`-0.0823e-12`
4)	$3250{,}4 \cdot 10^{6}$	`3250.4e06`
5)	$3{,}421 \cdot 10^{16}$	`3.421e16`

Очевидно, что `8.12e-5`, `812e-7`, `0.812e-4`, `81.2e-6` представляют одно и то же значение $8{,}12 \cdot 10^{-5}$.

Вопросы и упражнения

❶ Какие из приведенных ниже примеров соответствуют определению лексической единицы <*Целое число*>?

a)	`-418`	*f)*	`0+2469`	*k)*	`32,014`
b)	`0-418`	*g)*	`32,14`	*l)*	`-719`
c)	`621+`	*h)*	`+00621`	*m)*	`+62.1`
d)	`2469`	*i)*	`24693.`	*n)*	`-00418`
e)	`-6210`	*j)*	`-621`	*o)*	`-00621`

Найдите целые числа с одним и тем же значением.

❷ С помощью синтаксических диаграмм *рис. 1.4* напишите формулы БНФ для определения лексической единицы <*Целое число*>.

❸ Какие из приведенных ниже примеров соответствуют определению лексической единицы <*Вещественное число*>?

a)	`3.14`	*h)*	`281.3`	*o)*	`0,618284e00`
b)	`2.514e+5`	*i)*	`591328`	*p)*	`1961.`
c)	`591328E+3`	*j)*	`2514e+2`	*q)*	`28130E-2`
d)	`.000382`	*k)*	`-464.597e+3`	*r)*	`591.328`
e)	`0.1961E+4`	*l)*	`+519.328e-4`	*s)*	`-658.14e-6`
f)	`+314629.`	*m)*	`591328e-3`	*t)*	`2514e+2`
g)	`0.000314E4`	*n)*	`28130e-2`	*u)*	`618.248e-3`

Найдите вещественные числа, которые имеют одно и то же значение. Запишите данные числа в обычном виде.

❹ С помощью синтаксических диаграмм *рис. 1.4* напишите формулы БНФ для определения лексической единицы <*Вещественное число*>.

❺ Укажите на синтаксических диаграммах *рис. 1.4* пути, которые соответствуют следующим числам:

a)	`-418`	*e)*	`1961.0`	*i)*	`2514E+2`
b)	`281.3`	*f)*	`32.014`	*j)*	`+0001`
c)	`2.514e+5`	*g)*	`591.328`	*k)*	`-614.85e-3`
d)	`-1951.12`	*h)*	`+19.511e+2`	*l)*	`2013e-4`

1.5.4. Строки символов

Строка символов представляет собой последовательность печатных символов, заключенных в одиночные кавычки. Если в состав строки символов нужно включить сам символ одиночной кавычки, то этот символ печатается два раза. Отметим, что в строках символов строчные и прописные буквы являются различными символами.

Например:

1) `'Variabila x'`
2) `'Calculul aproximativ'`
3) `'Apostroful '' este dublat'`

В отличие от других лексических единиц языка ПАСКАЛЬ, в строках символов могут использоваться и буквы румынского и русского алфавитов. Для этого необходимо, чтобы на компьютере была установлена **программа-драйвер**, обеспечивающая ввод, вывод и печать таких букв.

Пример:

1) `'Şir de caractere'`
2) `'Строка'`
3) `'Английский язык'`
4) `'Număr încercări'`
3) `'Число попыток'`
4) `'Поверхность'`

Лексическая единица <*Строка*> определяется с помощью следующих формул БНФ:

<*Строка*> ::= ' <*Элемент строки*> {<*Элемент строки*>} '

<*Элемент строки*> ::= '' | <*Любой печатный символ*>

Синтаксическая диаграмма лексической единицы <*Строка*> показана на *рис. 1.5*.

Вопросы и упражнения

❶ Укажите на синтаксических диаграммах *рис. 1.5* пути, которые соответствуют следующим строкам:

a) `'переменная z'`

b) `''''`

c) `'Символы ''x'', ''y'''`

d) `'Лексические единицы'`

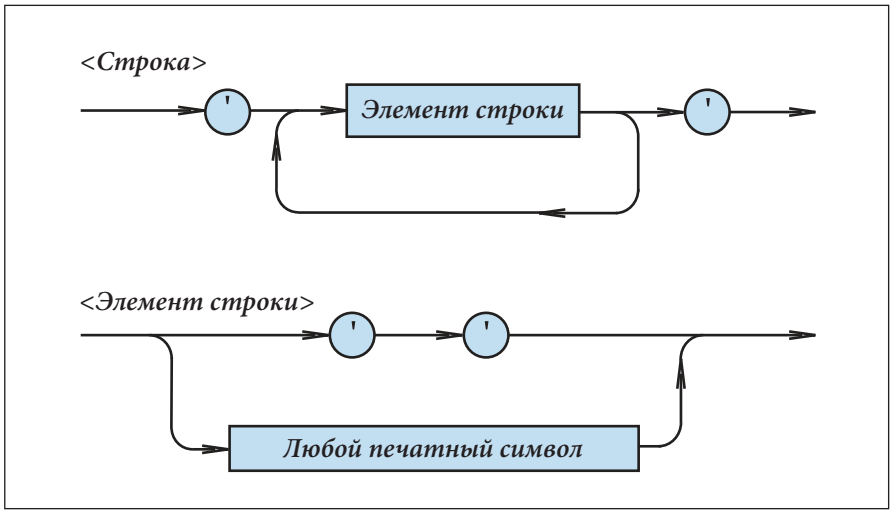


Рис. 1.5. Синтаксическая диаграмма <*Строка*>

❷ Какие из приведенных ниже примеров соответствуют определению лексической единицы <*Строка*>:

a) `'Целое число'`

b) `'Конец программы'`

c) `'АПОСТРОФ'`

d) `''x''`

e) `'функция'`

f) `'Год 1997'`

g) `'Квадратный корень'`

h) `'Год '97'`

i) `'Список телефонов'`

j) `'''`

1.5.5. Метки

Метка представляет собой целое число без знака в диапазоне от `0` до `9999`, которое используется для того, чтобы ссылаться на операторы языка ПАСКАЛЬ.

Примеры:

```
1    100    999    582    1004
```

Очевидно, что формула БНФ, которая определяет данную лексическую единицу, имеет вид:

<*Метка*> ::= <*Целое без знака*>

1.5.6. Директивы

Директивы – это зарезервированные слова, которые имеют специальное назначение. Лексическая единица <*Директива*> определяется точно так же, как идентификатор:

<*Директива*> ::= <*Буква*> {<*Буква*> | <*Цифра*>}

Стандартный язык содержит одну-единственную директиву:

```
forward
```

Данная директива используется при описании некоторых процедур и функций, определенных пользователем.

1.6. Разделители

Любая программа на языке ПАСКАЛЬ состоит из лексических единиц и разделителей. В языке ПАСКАЛЬ в качестве разделителей используются: *пробел*, *конец строки* (*возврат каретки*) и *комментарий*.

Примеры:

1)
```
x div y
```

2)
```
not x
```

3)
```
begin
  writeln(x);
  writeln(y);
end.
```

При отсутствии разделителей, при последовательном написании идентификаторов, ключевых слов, чисел без знака и директив, начало новой лексической единицы может быть интерпретировано как продолжение предыдущей.

В частности, в первом примере запись "`x div y`" сообщает компьютеру "раздели переменную `x` на переменную `y`". При отсутствии же разделительных пробелов, запись "`xdivy`" будет воспринята компьютером как идентификатор.

Отметим, что специальные символы, состоящие из двух знаков: `<=`, `>=`, `<>`, `:=`, `..` , идентификаторы, числа и т. д. являются лексическими единицами программы. Очевидно, вставка пробелов или символов возврата каретки внутри составных лексическх единиц недопустима.

Примеры:

	Правильно	*Неправильно*
1)	`CitireDisc`	`Citire Disc`
2)	`Program`	`Pro gram`
3)	`:=`	`: =`
4)	`..`	`. .`
5)	`345`	`3 45`
6)	`downto`	`down to`

7) `begin` `be gin`

Любая последовательность символов, заключенная в фигурные скобки {}, является **комментарием.**

Примеры:

1) `{ Программа, разработанная Александром Ивановым }`

2) `{ Введение исходных данных }`

3)
```
{ Исходные данные вводятся с клавиатуры.
  Результаты будут выведены на экран и
  распечатаны на принтере через 3-4 минуты }
```

Комментарии никак не влияют на работу программ на языке ПАСКАЛЬ, а используются для уточнений, пояснений, дополнительной информации и т.д. Они предназначены для лиц, читающих соответствующую программу.

Отметим, что рациональное использование комментариев, пробелов и символов возврата каретки делает программы удобными для чтения.

Тест для самопроверки № 1

1. Синтаксис языка программирования исполнителя ***Робот*** описан с помощью следующих металингвистических формул:

<*Цифра*> ::= `0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`

<*Число*> ::= <*Цифра*> {<*Цифра*>}

<*Команда*> ::= `вверх | вниз | вправо | влево`

<*Оператор*> ::= <*Команда*>`(`<*Число*>`)`

<*Программа*> ::= **начало** {< *Оператор* >`;`} **конец**

Укажите синтаксически правильные программы:

a) `начало вверх(1); вправо(4); вниз(0); влево(00); конец`

b) `начало вверх(1); вправо(73); вниз(0); влево (00+23); конец`

c) `начало вниз(30); вправо(45); вверх(980); конец`

d) `начало влево(21); вниз(50); вправо(45); вверх(980); конец`

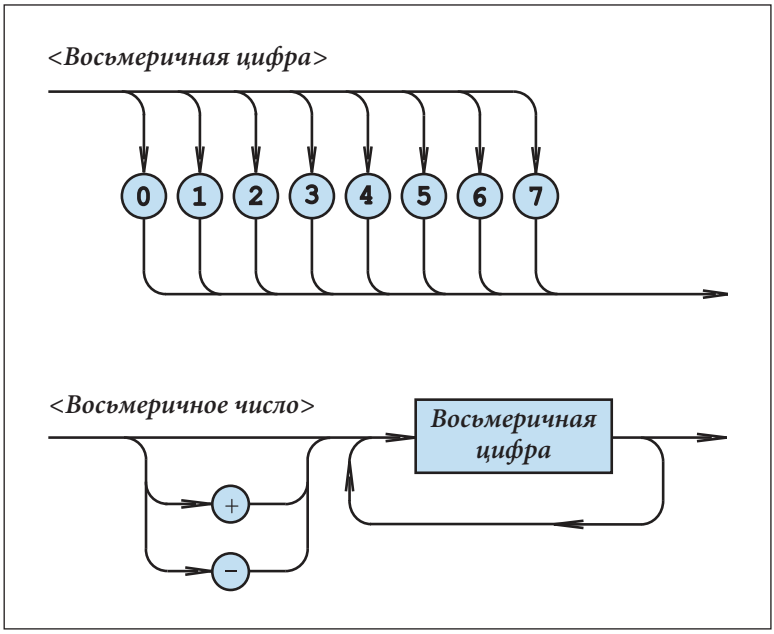
e) `начало влево(3); вниз(13); влево(21) конец; конец`

f) `начало вниз(73); вправо(5); вверх(71) влево(13); конец`

g) `начало вверх(1); вправо(-4); вниз(0); влево(10950); конец`

2. Нарисуйте синтаксические диаграммы, соответствующие металингвистическим формулам <*Команда*>, <*Оператор*> и <*Программа*> из задания 1.

3. На нижеследующем рисунке представлены синтаксические диаграммы, определяющие грамматическую единицу <*Восьмеричное число*>.



Определите, какие из приведенных ниже последовательностей символов записаны в соответствии с данными диаграммы <*Восьмеричное число*>:

a)	`+0`	*f)*	`-34637`	*k)*	`+123146482351`
b)	`18`	*g)*	`2347-523`	*l)*	`614,45`
c)	`-17250`	*h)*	`-0000007`	*m)*	`–152`
d)	`+6362,1`	*i)*	`527345372`	*n)*	`+35,1`
e)	`717424410571`	*j)*	`614.45`	*o)*	`-412`

4. Запишите металингвистические формулы, соответствующие синтаксическим диаграммам из задания 3.

5. На языке ПАСКАЛЬ идентификатор начинается с буквы, за которой может следовать любая комбинация из букв и цифр. Напишите металингвистические формулы, определяющие лексическую единицу <*Идентификатор*>.

6. Придумайте и запишите, по крайней мере, десять идентификаторов, отражающих специфику физических, математических, химических задач, а также задач по обработке текстов и изображений.

7. Запишите в соответствии с правилами языка ПАСКАЛЬ следующие, представленные в обычном виде, числа:

a)	3,14	*c)*	23,4635	e)	$6{,}1532 \cdot 10^{-5}$
b)	265	*d)*	+0,000001	*f)*	–984,52

g)	–523	j)	$614{,}45 \cdot 10^{-12}$	m)	$35728{,}345452 \cdot 10^{-8}$
h)	+28	k)	$-3628{,}297 \cdot 10^{12}$	n)	24815
i)	+28000000	l)	–38,00001	o)	–296,0020001

8. Запишите в обычном виде следующие числа, представленные в соответствии с правилами языка ПАСКАЛЬ:

a)	6124.485	f)	-0.03428e-08	k)	2005
b)	+18.315	g)	232847.5213	l)	+23.08e-5
c)	-218.034e-3	h)	-0000012e+2	m)	–17502
d)	193526	i)	18.45	n)	+1
e)	1000.01e+23	j)	623.495e-6	o)	-46341.2e-06

9. Известно, что словарь языка ПАСКАЛЬ включает следующие лексические единицы: специальные символы и ключевые слова, идентификаторы, числа, строки символов, метки, директивы. Укажите лексические единицы присутствующие в приведенной ниже программе:

```
1  Program TA1;
2  var a, b, x : real;
3  begin
4    readln(a, b);
5    if a<>0 then
6      begin
7        x:=-b/a;
8        writeln('Уравнение имеет единственный корень');
9        writeln(x);
10     end;
11   if (a=0) and (b=0) then
12       writeln( 'Уравнение имеет бесконечное
                   множество корней');
13   if (a=0) and (b<>0) then
14       writeln('Множество корней уравнения пусто');
15 end.
```

Напомним, что числа *1, 2, 3, ..., 15* в левой части страницы служат лишь для удобства при указании нужных строк и не являются частью программы на языке ПАСКАЛЬ.

Пример: **Program** – ключевое слово; TA1 – идентификатор; ; – специальный символ; **var** – ключевое слово и т.д.

10. Укажите заголовок, раздел описаний и раздел операторов программы TC1 из задания 9.

Глава 2

ПРОСТЫЕ ТИПЫ ДАННЫХ

2.1. Концепция данных

Информация, подлежащая обработке, представлена в компьютере в виде данных. **Данные** состоят из цифр, букв, знаков, чисел, строк и др.

В машинном коде компьютера данные представляются как последовательности двоичных цифр. Например, на языке процессора натуральное число 1039 представляется в двоичной системе счисления как:

```
10000001111
```

Для того чтобы избавить пользователя от деталей внутреннего представления данных, в языке ПАСКАЛЬ используются *типы данных*.

Под **типом данных** понимается **множество значений** и **множество операций**, которые можно к ним применять.

Например, в версии Turbo PASCAL 7.0 тип `integer` включает множество целых чисел

```
{-32768, -32767, ..., -2, -1, 0, 1, 2, ..., 32767}.
```

К данным числам можно применять следующие операции:

`+` сложение;
`-` вычитание;
`*` умножение;
`mod` остаток от деления;
`div` целочисленное деление и др.

Тип данных `real` (вещественный) включает подмножество множества вещественных чисел, с которыми выполняются операции `+`, `-`, `*`, `/` (деление) и др.

Операции **`mod`** и **`div`**, допустимые в случае данных типа `integer`, недопустимы в случае данных типа `real`.

В программах на языке ПАСКАЛЬ данные представляются в виде **величин**: переменных и констант. Термин «величина» позаимствован из математики и физики, где величины используются для описания природных явлений. В качестве примера, приведем некоторые величины, изучаемые на соответствующих уроках, это: масса m, длина l, площадь S, объем V, ускорение свободного падения $g \approx 9{,}8\ m/s^2$, иррациональное число $\pi \approx 3{,}14$ и др.

Переменная – это величина, значение которой может быть изменено в процессе выполнения программы. Любая переменная имеет имя, значение и тип. Имя переменной (например, `m`, `l`, `S`, `V`, `delta`) используется для ее обозначения в программе. В ходе выполнения программы каждая переменная, в текущий момент времени, имеет либо конкретное значение (например, 105 или -36), либо ее значение не определено.

Множество значений, которые может принимать переменная, и операции, допустимые для выполнения над ней, определяются посредством сопоставления имени переменной с необходимым типом данных. Для этого имя переменной и необходимый тип данных объявляются явным образом с помощью ключевого слова **var**.

Пример:

```
var x, y : integer;
    z : real;
```

В процессе выполнения программы переменные x и y могут принимать любые значения типа integer, а переменная z – любые значения типа real.

Константа – это величина, значение которой не может быть изменено в процессе выполнения программы. Тип константы объявляется неявно (по умолчанию) – через форму ее записи. Например, 10 – это константа целого типа, а 10.0 – константа вещественного типа.

Для большей наглядности константы могут иметь символические имена. Соответствующие имена определяются с помощью ключевого слова **const**.

Пример:

```
const g = 9.8;
      pi = 3.14;
```

Очевидно, тип констант g и pi является вещественным и их значения не могут быть изменены в ходе выполнения программы.

Концепция данных, реализованная в языке ПАСКАЛЬ, предполагает следующее:

1) каждая величина (переменная или константа), используемая в программе, должна иметь определенный тип данных;

2) тип переменной определяет множество значений, которые она может принимать, и операции, которые можно применять к данным значениям;

3) существуют предопределенные типы данных, которые считаются известными в любой программе: integer, real, char (символьный), boolean (логический), text (текстовый) и др.;

4) на основании известных типов данных программист может создавать новые типы, отражающие природу информации, подлежащей обработке.

Вопросы и упражнения

❶ Как представляются данные в машинном коде компьютера? Укажите преимущества и недостатки такого представления.

❷ Как представляются данные в программах на языке ПАСКАЛЬ? Укажите разницу между переменной и константой.

❸ Объясните значение термина *тип данных*. Приведите примеры.

❹ Как указывается тип переменных?

❺ Определите тип переменных r, s, t, x, y и z из следующего описания:

```
var r, y : integer;
    s, z : real;
    t, x : boolean;
```

❻ Опишите переменные a, b и c как переменные целого типа, а p и q как переменные текстового типа.

❼ Определите тип следующих констант:

a)	`-301`	*d)*	`-61.00e+2`	*g)*	`314.0`
b)	`-301.0`	*e)*	`3.14`	*h)*	`0314`
c)	`+6100`	*f)*	`-0.0001`	*i)*	`-0.000672`

2.2. Тип данных integer

Множество значений типа данных integer состоит из элементов подмножества целых чисел, определяемого при реализации. Максимальное значение определяется константой MaxInt, известной любой программе на языке ПАСКАЛЬ. Обычно минимальным значением в изучаемом типе данных является – MaxInt или – (MaxInt+1).

Приведенная ниже программа выводит на экран значение предописанной константы MaxInt.

```
Program P2;
 { Вывод предописанной константы MaxInt }
begin
  writeln('MaxInt=', MaxInt);
end.
```

В компьютерах IBM PC, работающих с версией Turbo PASCAL 7.0, константа MaxInt имеет значение 32767, а множеством значений типа integer является:

```
{-32768, -32767, ..., -2, -1, 0, 1, 2, ..., 32767}.
```

К целым значениям можно применять следующие операции: +, –, *, **mod** (остаток от деления), **div** (целая часть от деления) и др. Результаты данных операций можно увидеть с помощью программы P3.

```
Program P3;
 { Операции с данными типа integer }
var x, y, z : integer;
begin
  writeln('Введите целые числа x, y:');
  readln(x, y);
  writeln('x=', x);
  writeln('y=', y);
  z:=x+y; writeln('x+y=', z);
  z:=x-y; writeln('x-y=', z);
  z:=x*y; writeln('x*y=', z);
  z:=x mod y; writeln('x mod y=', z);
  z:=x div y; writeln('x div y=', z);
end.
```

Очевидно, что результаты операций +, –, * над целыми числами также должны принадлежать множеству значений типа данных `integer`. Если программист нарушает данное правило, появляются ошибки переполнения. Эти ошибки будут обнаружены в процессе компиляции или при выполнении соответствующей программы. Для пояснения приведем программы P4 и P5:

```
Program P4;
 {Ошибка переполнения, обнаруженная в процессе компиляции}
var x : integer;
begin
  x:=MaxInt+1; { Ошибка, x>MaxInt }
  writeln(x);
end.
```

```
Program P5;
 { Ошибка переполнения, обнаруженная в процессе выполнения}
var x, y : integer;
begin
  x:=MaxInt;
  y:=x+1; { Ошибка, y>MaxInt }
  writeln(y);
end.
```

Приоритеты операций +, -, *, **mod**, **div** будут изучены позже (*таблица 3.2*).

Вопросы и упражнения

❶ Укажите множество значений типа данных `integer`. Какие операции можно применять к данным значениям?

❷ Когда появляются ошибки переполнения? Как обнаруживаются такие ошибки?

❸ Найдите значение константы `MaxInt` версии языка ПАСКАЛЬ, с которой вы работаете.

❹ Даны следующие программы:

```
Program P6;
 { Ошибка переполнения }
var x : integer;
begin
  x:=-2*MaxInt;
  writeln(x);
end.
```

```
Program P7;
 { Ошибка переполнения }
var x, y : integer;
begin
  x:=-MaxInt;
```

```
  y:=x-10;
  writeln(y);
end.
```

На каком этапе будут обнаружены ошибки переполнения: на этапе компиляции или выполнения?

❺ Приведите примеры значений переменных `x` и `y` из программы `P3`, для которых появятся ошибки переполнения.

2.3. Тип данных `real`

Множество значений типа данных `real` состоит из элементов подмножества вещественных чисел, определяемого при реализации.

Например, в версии Turbo PASCAL 7.0 областью допустимых значений вещественного типа является $-1{,}7\cdot10^{38}$, ..., $+1{,}7\cdot10^{38}$, а числа могут быть указаны с точностью до 11–12 десятичных цифр.

В следующей программе переменным `x`, `y` и `z` присваиваются соответственно значения 1,1, $-6{,}14\cdot10^{8}$ и $90{,}3\cdot10^{-29}$, которые затем выводятся на экран.

```
Program P8;
 { Данные вещественного типа }
var x, y, z : real;
begin
  x:=1.1;
  y:=-6.14e8;
  z:=90.3e-29;
  writeln('x=', x);
  writeln('y=', y);
  writeln('z=', z);
end.
```

Напомним, что при записи вещественных чисел десятичная запятая заменяется точкой, а степень числа 10 представляется масштабным множителем (см. параграф 1.5.3).

Операции, которые можно применять к значениям вещественного типа: `+`, `–`, `*`, `/` (деление) и др.

Операции над вещественными числами в общем случае являются приближенными из-за погрешностей округления. Естественно, результаты данных операций также должны принадлежать множеству значений типа `real`. В противном случае возникают ошибки переполнения.

Свойства операций `+`, `–`, `*` и `/` могут быть изучены с помощью следующей программы:

```
Program P9;
 { Операции с данными типа real }
var x, y, z : real;
```

```
begin
  writeln('Введите вещественные числа x, y:');
  readln(x,y);
  writeln('x=', x);
  writeln('y=', y);
  z:=x+y; writeln('x+y=', z);
  z:=x-y; writeln('x-y=', z);
  z:=x*y; writeln('x*y=', z);
  z:=x/y; writeln('x/y=', z);
end.
```

В *таблице 2.1* представлены данные, которые выводятся на экран программой P9 (версия Turbo PASCAL 7.0) для некоторых значений переменных x и y. Отметим, что результаты операций $x+y$ и $x-y$ в первых двух строках *таблицы 2.1* являются точными. Для значений $x = 1{,}0$, $y = 1{,}0 \cdot 10^{-11}$ (третья строка таблицы) результат сложения – приближенный, а вычитания – точный. Оба результата из строки 4 являются приближенными. Для значений $x = y = 1{,}7 \cdot 10^{38}$ (строка 5) при выполнении операции сложения имеет место переполнение. Для значений $x = 3{,}1 \cdot 10^{-39}$, $y = 3{,}0 \cdot 10^{-39}$ (строка 6) результат сложения – точный, а вычитания – приближенный.

Таблица 2.1.

Результаты работы программы P9

№ пп.	x	y	$x + y$	$x - y$
1	1,0	1,0	2.0000000000E+00	0.0000000000E+00
2	1,0	$1{,}0 \cdot 10^{-10}$	1.0000000000E+00	9.9999999999E-01
3	1,0	$1{,}0 \cdot 10^{-11}$	1.0000000000E+00	9.9999999999E-01
4	1,0	$1{,}0 \cdot 10^{-12}$	1.0000000000E+00	1.0000000000E+00
5	$1{,}7 \cdot 10^{38}$	$1{,}7 \cdot 10^{38}$	переполнение	0.0000000000E+00
6	$3{,}1 \cdot 10^{-39}$	$3{,}0 \cdot 10^{-39}$	6.1000000000E-39	0.0000000000E+00

Ошибки вычисления, характерные для типа данных `real`, могут нарушить ход выполнения программы. Оценка ошибок и, если необходимо, их устранение являются задачей программиста.

Приоритеты операций +, -, *, / будут изучены позже (*таблица 3.2*).

Вопросы и упражнения

❶ Как записываются вещественные числа на языке ПАСКАЛЬ?

❷ Определите множество значений вещественного типа данных в версии языка ПАСКАЛЬ, в которой вы работаете. Какова точность соответствующих чисел?

❸ Какие операции можно применять к данным типа `real`? Являются ли эти операции точными?

❹ Запустите на выполнение программу P9 для следующих значений переменных x, y:

a) $x = 2{,}0$; $y = -3{,}0$; *b)* $x = 14{,}3 \cdot 10^{2}$; $y = 15{,}3 \cdot 10^{-3}$;

c) $x = 3{,}0;$ $y = 2{,}0 \cdot 10^{12};$

d) $x = 3{,}0;$ $y = 2{,}0 \cdot 10^{-12};$

e) $x = 2{,}9 \cdot 10^{-39};$ $y = 6{,}4 \cdot 10^{-3};$

f) $x = 7{,}51 \cdot 10^{21};$ $y = -8{,}64 \cdot 10^{17};$

g) $x = 1{,}0;$ $y = 2{,}9 \cdot 10^{-39};$

h) $x = 1{,}7 \cdot 10^{38};$ $y = 2{,}9 \cdot 10^{-39}.$

Проверьте результаты соответствующих операций. Объясните сообщения, выдаваемые на экран.

❺ Каковы причины ошибок вычисления при работе с данными типа `real`?

2.4. Тип данных `boolean`

Значениями типа данных `boolean` (логический) являются истинностные значения `false` (ложь) и `true` (истина). В ниже приведенной программе переменной `x` последовательно присваиваются значения `false` и `true`, выводимые впоследствии на экран.

```
Program P10;
 { Данные типа boolean }
var x : boolean;
begin
 x:=false;
 writeln(x);
 x:=true;
 writeln(x);
end.
```

Логические операции, которые можно применять к данным логического типа:

not	отрицание (логическая инверсия, логическая операция НЕ);
and	конъюнкция (логическое произведение, логическая операция И);
or	дизъюнкция (логическая сумма, логическая операция ИЛИ).

Таблицы истинности данных операций представлены на *рис. 2.1*.

Свойства логических операций **not**, **and** и **or** могут быть изучены с помощью программы `P11`.

```
Program P11;
 { Операции с данными типа boolean }
var x, y, z : boolean;
begin
 x:=false; y:=false;
 writeln('x=', x, 'y=', y);
 z:=not x; writeln('not x = ', z);
 z:=x and y; writeln('x and y = ', z);
 z:=x or y; writeln('x or y = ', z);
 writeln;
 x:=false; y:=true;
```

```
  writeln('x=', x, 'y=', y);
  z:=not x; writeln('not x = ', z);
  z:=x and y; writeln('x and y = ', z);
  z:=x or y; writeln('x or y = ', z);
  writeln;
  x:=true; y:=false;
  writeln('x=', x, 'y=', y);
  z:=not x; writeln('not x = ', z);
  z:=x and y; writeln('x and y = ', z);
  z:=x or y; writeln('x or y = ', z);
  writeln;
  x:=true; y:=true;
  writeln('x=', x, 'y=', y);
  z:=not x; writeln('not x = ', z);
  z:=x and y; writeln('x and y = ', z);
  z:=x or y; writeln('x or y = ', z);
  writeln;
end.
```

x	not x
false	true
true	false

x	y	x and y
false	false	false
false	true	false
true	false	false
true	true	true

x	y	x or y
false	false	false
false	true	true
true	false	true
true	true	true

Рис. 2.1. Таблицы истинности логических операций **not**, **and** и **or**

В отличие от переменных целого или вещественного типа текущие значения переменных логического типа не могут считываться с клавиатуры с помощью стандартной процедуры `readln`. Поэтому в программе `P11` текущие значения переменных `x` и `y` задаются через присваивание.

Приоритеты операций **`not`**, **`and`**, **`or`** будут изучены позже (*таблица 3.2*).

Вопросы и упражнения

❶ Назовите множество значений данных логического типа и операции, применимые к ним.
❷ Выучите таблицы истинности логических операций.
❸ Напишите программу, которая выводит на экран таблицу истинности логической операции **`not`**.
❹ Напишите программу, которая вычисляет значения логической функции $z = x \,\&\, y$ для всевозможных значений аргументов x, y.
❺ Напишите программу, которая выводит на экран значения логической функции $z = x \vee y$.

2.5. Тип данных `char`

Множеством значений данного типа является конечное упорядоченное множество символов. Значения рассматриваемого типа обозначаются символом, заключенным в одиночные кавычки (апострофы), например: `'A'`, `'B'`, `'C'` и т.д. Если нужен сам символ апострофа, то он дублируется: `''''`.

В следующей программе переменной `x` типа `char` последовательно присваиваются значения `'A'`, `'+'` и `''''`, выводимые на экран.

```
Program P12;
 { Данные типа char }
var x : char;
begin
 x:='A';
 writeln(x);
 x:='+';
 writeln(x);
 x:='''';
 writeln(x);
end.
```

Текущие значения переменной типа `char` могут считываться с клавиатуры с помощью стандартной процедуры `readln`. Для пояснения представляем программу `P13`, которая считывает с клавиатуры и выводит на экран значения типа `char`.

```
Program P13;
 { Считывание и вывод символов на экран }
var x : char;
```

```
begin
 readln(x); writeln(x);
 readln(x); writeln(x);
 readln(x); writeln(x);
end.
```

Соответствующие символы вводятся с клавиатуры и выводятся на экран без апострофов, которые необходимы лишь для включения символьных значений в текст программы.

Как правило, символы конкретной версии языка ПАСКАЛЬ **упорядочены** согласно таблице кодов *ASCII* (см. параграф 1.4).

Порядковый номер символа в упорядоченном множестве значений типа `char` можно найти с помощью стандартной функции `ord`, например:

1) `ord('A')= 65`

2) `ord('B')= 66`

3) `ord('C')= 67`

и т.д.

Следующая программа выводит на экран порядковые номера любых четырех символов, считываемых с клавиатуры.

```
Program P14;
 { Изучение функции ord }
var x : char; { символ }
    i : integer; { порядковый номер }
begin
 readln(x); i:=ord(x); writeln(i);
 readln(x); i:=ord(x); writeln(i);
 readln(x); i:=ord(x); writeln(i);
 readln(x); i:=ord(x); writeln(i);
end.
```

Стандартная функция `chr` возвращает символ, который соответствует указанному порядковому номеру. Таким образом,

1) `chr(65)='A';`

2) `chr(66)='B';`

3) `chr(67)='C'`

и т.д.

Программа `P15` выводит на экран символы, которые соответствуют порядковым номерам, считанным с клавиатуры.

```
Program P15;
 { Изучение функции chr }
var i : integer; { порядковый номер }
```

```
    x : char; { символ }
begin
 readln(i); x:=chr(i); writeln(x);
 readln(i); x:=chr(i); writeln(x);
 readln(i); x:=chr(i); writeln(x);
 readln(i); x:=chr(i); writeln(x);
end.
```

Напоминаем, что расширенное множество *ASCII* включает 256 пронумерованных символов: 0, 1, 2, …, 255.

Тип данных `char` используется для создания более сложных структур данных, в частности строк символов.

Вопросы и упражнения

❶ Укажите множество значений типа данных `char`.
❷ Как упорядочено множество значений типа `char`?
❸ Определите порядковые номера следующих символов:
– десятичных цифр;
– прописных букв латинского алфавита;
– знаков препинания;
– знаков арифметических и логических операций;
– управляющих символов;
– букв русского алфавита (если они установлены на вашем компьютере).
❹ Определите символы, которые соответствуют следующим порядковым номерам:

77	109	79	111	42	56	91	123

❺ Напишите программу, которая выводит на экран множество всех символов компьютера, на котором вы работаете.

2.6. Перечисляемые типы данных

Типы `integer`, `real`, `boolean` и `char`, изучаемые до сих пор, являются предопределенными типами данных, известными любой программе на языке ПАСКАЛЬ. Кроме предопределенных типов данных существуют типы, создаваемые самим программистом, в частности *перечисляемые* типы.

Перечисляемый тип данных содержит упорядоченное множество значений, определяемых посредством идентификаторов. Имя и множество значений *перечисляемого* типа данных указываются в разделе описаний программы после ключевого слова **`type`** (тип).

Пример:

```
type Culoare = (Galben, Verde, Albastru, Violet);
     Studii = (Elementare, Medii, Superioare);
     Raspuns = (Nu, Da);
```

Первый идентификатор из списка элементов является наименьшим значением и его порядковый номер равен нулю. Второй идентификатор имеет порядковый номер один, третий – два и т.д. Порядковый номер любого перечисляемого значения может быть найден с помощью стандартной функции `ord`.

Примеры:

1) `ord(Galben)= 0`
2) `ord(Verde)= 1`
3) `ord(Albastru)= 2`
4) `ord(Violet)= 3`
5) `ord(Elementare)= 0`
6) `ord(Medii)= 1`

и т.д.

Следующая программа выводит на экран порядковые номера значений типа данных `Studii`.

```
Program P16;
 { Тип данных Studii }
type Studii = (Elementare, Medii, Superioare);
var i : integer; { numar de ordine }
begin
 i:=ord(Elementare); writeln(i);
 i:=ord(Medii); writeln(i);
 i:=ord(Superioare);
 writeln(i);
end.
```

Переменные *перечисляемого* типа описываются с помощью ключевого слова **`var`**. Они могут принимать только значения перечисляемых элементов из типа данных, к которому они относятся.

В следующей программе переменная `x` принимает значение `Albastru`, а переменная `y` принимает значение `Nu`. Порядковые номера этих значений выводятся на экран.

```
Program P17;
 { Переменные перечисляемого типа }
type Culoare = (Galben, Verde, Albastru, Violet);
     Raspuns = (Nu, Da);
var  x : Culoare; { Переменная типа Culoare }
     y : Raspuns; { Переменная типа Raspuns }
     i : integer; { порядковый номер }
begin
 x:=Albastru;
 i:=ord(x); writeln(i);
 y:=Nu; i:=ord(y); writeln(i);
end.
```

В случаях, когда в одной программе описываются несколько типов данных, соответствующие списки элементов не должны содержать одинаковые идентификаторы.

Например, объявление

```
type Studii = (Elementare, Medii, Superioare);
     Grade = (Inferioare, Superioare)
```

является неправильным, так как идентификатор `Superioare` появляется в обоих списках.

Текущие значения переменных *перечисляемых* типов не могут считываться с клавиатуры и выводиться на экран с помощью стандартных процедур `readln` и `writeln`. Однако использование таких типов данных позволяет создавать простые и эффективные программы, удобные для чтения.

Вопросы и упражнения

❶ Как определяется *перечисляемый* тип данных? Укажите множество значений *перечисляемого* типа данных.

❷ Существенен ли порядок идентификаторов в списке элементов *перечисляемого* типа данных?

❸ Напишите программу, которая выводит на экран порядковые номера значений следующих типов данных:

a)
```
Continente = (Europa, Asia, Africa, AmericaDeNord,
              AmericaDeSud, Australia, Antarctida);
```

b)
```
Genul = (Masculin, Feminin);
```

c)
```
PuncteCardinale = (Nord, Sud, Est, Vest);
```

d)
```
Etaje = (Unu, Doi, Trei, Patru, Cinci);
```

❹ Назовите тип каждой переменной, описанной в программе `P18`.

```
Program P18;
type Litere = (A, B, C, D, E, F, G);
var x : Litere; y : char; i : integer;
begin
 x:=A; i:=ord(x); writeln(i);
 y:='A'; i:=ord(y); writeln(i);
end.
```

Что выводит на экран указанная программа?

❺ Даны объявления:

```
type Culoare = (Galben, Verde, Albastru, Violet);
     Fundal = (Alb, Negru, Gri);
var  x, y : Culoare;
     z : Fundal;
```

Какие из следующих операторов являются правильными?

a) `x:=Verde`

b) `y:=Negru`

c) `z:=Alb`

d) `x:=Gri`

e) `y:=Gri`

f) `z:=Violet`

g) `x:=Albastru`

h) `y:=Azuriu`

❻ В программе `P17` перед ключевым словом **end** вставьте одну из следующих строк:

a) `readln(x);`

b) `writeln(x);`

Объясните сообщения, выводимые на экран в процессе компиляции полученной программы.

2.7. Интервальные типы данных

Интервальный тип данных включает подмножество значений уже известного типа данных, называемого базовым. *Базовым типом* может быть `integer`, `boolean`, `char` или любой *перечисляемый* тип.

Имя *интервального* типа данных, его наименьшее и наибольшее значение (в смысле порядкового номера) указываются в разделе описаний программы после ключевого слова **type**.

Например:

1)
```
type Indice = 1..10;
     Litera = 'A'..'Z';
     Cifra = '0'..'9';
```

Множество значений типа `Indice` является подмножеством значений предопределенного типа `integer`. Множества значений типов `Litera` и `Cifra` являются подмножествами предопределенного типа `char`.

2)
```
type Zi = (L, Ma, Mi, J, V, S, D);
     ZiDeLucru = L..V;
     ZiDeOdihna = S..D;
```

Множества значений типов `ZiDeLucru` и `ZiDeOdihna` являются подмножествами перечисляемого типа `Zi`, определенного пользователем.

3)
```
type T1 = (A, B, C, D, E, F, G, H);
     T2 = A..F;
     T3 = C..H;
```

Множества значений типов `T2` и `T3` – подмножества *перечисляемого* типа `T1`.

Из данных примеров следует, что базовыми типами *интервальных типов* данных являются:

	Интервальный тип	*Базовый тип*
1)	`Indice`	`integer`
2)	`Litera`	`char`
3)	`Cifra`	`char`

4)	`ZiDeLucru`	`Zi`
5)	`ZiDeOdihna`	`Zi`
6)	`T2`	`T1`
7)	`T3`	`T1`

Переменные *интервального типа* объявляются с помощью ключевого слова **`var`**. Переменная *интервального типа* обладает всеми свойствами переменных базового типа, но ее значения должны находиться в соответствующем диапазоне. В противном случае возникает ошибка и программа останавливается.

Пример:

```
Program P19;
 { Значения переменных интервального типа }
type Indice = 1..10;
     Zi = (L, Ma, Mi, J, V, S, D);
     ZiDeLucru = L..V;
     ZiDeOdihna = S..D;
var  i : Indice;  { возможные значения: 1, 2, ..., 10 }
     z : Zi;      { возможные значения: L, Ma, ..., D }
     zl : ZiDeLucru; { возможные значения: L, Ma, ..., V }
     zo : ZiDeOdihna; { возможные значения: S, D }
begin
 i:=5; i:=11;          { Ошибка, i>10 }
 z:=L; zl:=J; zl:=S;   { Ошибка, zl>V }
 zo:=S; zo:=V;         { Ошибка, zo<S }
 writeln('Конец');
end.
```

Программа `P20` показывает, как тип `Pozitiv` наследует свойства типа `integer`.

```
Program P20;
 { Тип Pozitiv наследует свойства типа integer }
type Pozitiv = 1..32767;
var x, y, z : Pozitiv;
begin
 writeln('Введите положительные числа x, y:');
 readln(x,y);
 writeln('x=', x);
 writeln('y=', y);
 z:=x+y; writeln('x+y=', z);
 z:=x-y; writeln('x-y=', z);
 z:=x*y; writeln('x*y=', z);
 z:=x mod y; writeln('x mod y=', z);
 z:=x div y; writeln('x div y=', z);
end.
```

Отметим, что операции +, -, *, **mod** и **div** базового типа integer применимы к *интервальному* типу Pozitiv. Но в отличие от переменных типа integer, переменные типа Pozitiv не могут принимать отрицательные значения.

Использование *интервальных* типов данных улучшает наглядность программ и упрощает их проверку. Следует подчеркнуть, что в языке ПАСКАЛЬ нельзя определять интервальные типы на базе типа real, так как его значения не имеют порядковых номеров.

Вопросы и упражнения

❶ Как определяется *интервальный* тип данных? Из чего состоит множество значений *интервального* типа данных?

❷ Назовите базовый тип каждого *интервального* типа:

```
type T1 = (A, B, C, D, E, F, G, H);
     T2 = -60..60;
     T3 = 5..9;
     T4 = '5'..'9';
     T5 = A..E;
     T6 = 'A'..'E';
```

❸ Какие значения может принимать каждая переменная из следующих описаний:

```
type T1 = (A, B, C, D, E, F, G, H);
     T2 = 1..9;
     T3 = 6..15;
     T4 = -100..100;
     T5 = 'A'..'Z';
     T6 = '0'..'9';
     T7 = C..F;
var  i : integer;
     j : T2;
     m : T4;
     p : T5;
     q : char;
     r : T6;
     s : T1;
     t : T7;
```

Назовите базовый тип каждого *интервального* типа. Укажите множество операций, унаследованных от базового типа.

❹ Какие из следующих определений являются правильными? Аргументируйте ваш ответ.

a)
```
type Lungime = 1.0e-2..1.0;
     Latime = 1.0e-2..0.5;
```

b)
```
type Indice = 1..10;
     Abatere = +5..-5;
     Deviere = -10..+10;
```

c)
```
type T1 = (A, B, C, D, E, F, G, H);
     T2 = C..H;
     T3 = F..B;
```

d)
```
type Luni = ( Ianuarie, Februarie, Martie, Aprilie, Mai,
              Iunie, Iulie, August, Septembrie,
              Octombrie, Noiembrie, Decembrie);
     LuniDeIarna = (Decembrie..Februarie);
     LuniDePrimavara = (Martie..Mai);
     LuniDeVara=(Iunie..August);
     LuniDeToamna=(Septembrie..Noiembrie);
```

❺ Дана следующая программа:

```
Program P21;
 type Indice=1..10;
 var  i, j, k, m : Indice;
begin
 writeln('Введите индексы i, j:');
 readln(i, j);
 k:=i+j; writeln('k=', k);
 m:=i-j; writeln('m=', m);
end.
```

При каких значениях переменных `i`, `j` появятся ошибки выполнения?

a) `i=3, j=2;`
b) `i=7, j=4;`
c) `i=4, j=7;`
d) `i=6, j=3;`
e) `i=2, j=2;`
f) `i=3, j=11;`
g) `i=8, j=4;`
h) `i=5, j=3.`

❻ Рассмотрим программу `P20`. После запуска на выполнение пользователь вводит `x=1, y=2.` Очевидно, что `x-y=-1.` Так как значение `-1` не принадлежит типу данных `Pozitiv`, то при выполнении оператора

```
z:=x-y
```

возникнет ошибка.
Укажите команды, при выполнении которых появятся ошибки, если:

a) `x=1000, y=1000;`
b) `x=1000, y=1001;`
c) `x=1001, y=1000;`
d) `x=30000, y=30000;`
e) `x=1, y=2;`
f) `x=1000, y=100;`
g) `x=0, y=1;`
h) `x=1, y=0.`

2.8. Порядковые типы данных

Типы данных `integer`, `boolean`, `char`, *перечисляемые* и *интервальные* типы являются **порядковыми типами данных**. Каждое значение порядкового типа данных имеет порядковый номер, определяемый следующим образом:

1) порядковым номером некоторого числа типа `integer` является само это число;

2) порядковыми номерами истинностных значений `false` и `true` типа данных `boolean` являются соответственно `0` и `1`;

3) порядковый номер некоторого символа (тип `char`) определяется его позицией в таблице кодов, обычно *ASCII*;

4) порядковый номер некоторого значения *перечисляемого* типа определяется его позицией в списке перечисляемых элементов. Отметим, что элементы списка нумеруются начиная с нуля: 0, 1, 2, … и т.д.

5) порядковые номера значений *интервального* типа совпадают с их порядковыми номерами в базовом типе.

Порядковый номер любого значения порядкового типа можно узнать с помощью стандартной функции `ord`.

Программа `P22` выводит на экран порядковые номера значений: `-32`, `true`, `'A'`, `A` и `B`.

```
Program P22;
 { Порядковые номера значений порядкового типа }
type T1 = (A, B, C, D, E, F, G, H);
     T2 = B..G;
begin
 writeln(ord(-32));    { -32 }
 writeln(ord(true));   { 1 }
 writeln(ord('A'));    { 65 }
 writeln(ord(A));      { 0 }
 writeln(ord(B));      { 1 }
end.
```

К значениям любого порядкового типа можно применять операции отношения:

`<`	меньше;
`<=`	меньше или равно;
`=`	равно;
`>=`	больше или равно;
`>`	больше;
`<>`	не равно.

Результатом операции отношения является значение типа `boolean`: `true` или `false`. При выполнении таких операций сравниваются не сами значения, а их порядковые номера.

Например, при следующем описании:

```
type Culoare = (Galben, Verde, Albastru, Violet);
```

результатом операции

```
Verde < Violet
```

является `true`, так как `ord(Verde) = 1`, `ord(Violet)=3`, а 1 меньше чем 3.

Результатом операции

```
Galben > Violet
```

является `false`, так как `ord(Galben)=0, ord(Violet)=3,` а 0 не больше чем 3.

Следующая программа выводит на экран результаты операций отношения для значений `Verde` и `Violet` типа данных `Culoare`.

```
Program P23;
 { Операции отношения над значениями порядкового типа }
type Culoare = (Galben, Verde, Albastru, Violet);
begin
 writeln(Verde<Violet);    { true }
 writeln(Verde<=Violet);   { true }
 writeln(Verde=Violet);    { false }
 writeln(Verde>=Violet);   { false }
 writeln(Verde>Violet);    { false }
 writeln(Verde<>Violet);   { true }
end.
```

Для порядковых типов данных существуют стандартные функции `pred` (*предшествующий*) и `succ` (*последующий*).

Значению с порядковым номером *i* предшествует значение с порядковым номером *i*–1. За значением с порядковым номером *i* следует значение с порядковым номером *i*+1.

Например, для значений порядкового типа данных `Culoare` получаем:

```
pred(Verde) = Galben;
```

```
succ(Verde) = Albastru;
```

```
pred(Albastru) = Verde;
```

```
succ(Albastru) = Violet.
```

Очевидно, что наименьшее значение не имеет предшествующего, а наибольшее – последующего.

Программа `P24` выводит на экран значения, предшествующие и следующие за `'B'`, `0` и `'0'`.

```
Program P24;
 { Предшествующие и следующие значения }
begin
 writeln(pred('B'));   { 'A' }
 writeln(succ('B'));   { 'C' }
 writeln(pred(0));     { -1 }
 writeln(succ(0));     { 1 }
 writeln(pred('0'));   { '/' }
 writeln(succ('0'));   { '1' }
end.
```

Отметим, что значения, предшествующие (или следующие) за порядковыми значениями `0` (тип `integer`) и `'0'` (тип `char`) не совпадают, так как типы этих значений различны.

Тип данных `real` не является порядковым типом. Следовательно, к вещественным значениям нельзя применять функции `ord` (возвращает порядковый номер), `pred` (возвращает предыдущее значение) и `succ` (возвращает последующее значение). Несоблюдение этого правила приводит к ошибкам.

Вопросы и упражнения

❶ Назовите порядковые типы данных. Какими общими свойствами они обладают?

❷ Как определяются порядковые номера значений любого порядкового типа данных?

❸ Что выведет на экран следующая программа?

```
Program P25;
type Zi = (L, Ma, Mi, J, V, S, D);
var z1, z2 : Zi;
begin
 z1:=Ma;
 writeln(ord(z1));
 z2:=pred(z1);
 writeln(ord(z2));
 z2:=succ(z1);
 writeln(ord(z2));
 z1:=Mi; z2:=V;
 writeln(z1<z2);
 writeln(z1>z2);
 writeln(z1<>z2);
end.
```

❹ Удалите из следующей программы строку, содержащую ошибку:

```
Program P26;
 { Ошибка }
var i : integer;
begin
 i:=MaxInt;
 writeln(pred(i));
 writeln(succ(i));
end.
```

Что будет выведено на экран после запуска модифицированной программы на выполнение?

❺ Прокомментируйте следующую программу:

```
Program P27;
 { Ошибка }
var i : integer; x : real;
begin
 i:=1; x:=1.0;
```

```
  writeln(ord(i));
  writeln(ord(x));
  writeln(pred(i));
  writeln(pred(x));
  writeln(succ(i));
  writeln(succ(x));
end.
```

Удалите строки, содержащие ошибки. Что будет выведено на экран после запуска модифицированной программы?

2.9. Объявление типов данных

Язык ПАСКАЛЬ предлагает пользователю предопределенные типы данных: `integer`, `real`, `boolean`, `char` и др. В случае необходимости пользователь может создавать собственные типы данных, например *перечисляемые* или *интервальные*.

Имя типа данных и множество его значений объявляются с помощью следующих грамматических единиц:

<*Раздел описания типов* > ::= **type** <*Описание типа*>; { <*Описание типа*>;}

<*Описание типа*> ::= <*Идентификатор*> = <*Тип*>

<*Тип*> ::= <*Идентификатор*> | <*Перечисляемый тип*> | <*Интервальный тип*>
| <*Тип-массив*> | <*Тип-запись*> | <*Тип-множество*> | <*Файловый тип*>
| <*Ссылочный тип*>

<*Перечисляемый тип*> ::= (<*Идентификатор*> {, <*Идентификатор*>})

<*Интервальный тип*> ::= <*Константа*>..<*Константа*>

Соответствующие синтаксические диаграммы приведены на *рис. 2.2.*

Примеры:

1)
```
type T1 = (A, B, C, D, E, F, G, H);
     T2 = B..F;
     T3 = C..H;
```

2)
```
type Pozitiv = 1..MaxInt;
     Natural = 0..MaxInt;
     Negativ = -MaxInt..-1;
```

3)
```
type Abatere = -10...+10;
     Litera = 'A'..'Z';
     Cifra = '0'..'9';
```

Классификация типов данных языка ПАСКАЛЬ представлена на *рис. 2.3*. Ранее изученные типы данных указаны на темном фоне.

В некоторых конструкциях языка ПАСКАЛЬ необходимо, чтобы переменные и константы принадлежали к идентичным или совместимым типам.

Два типа являются **идентичными**, если они задаются одним и тем же именем.

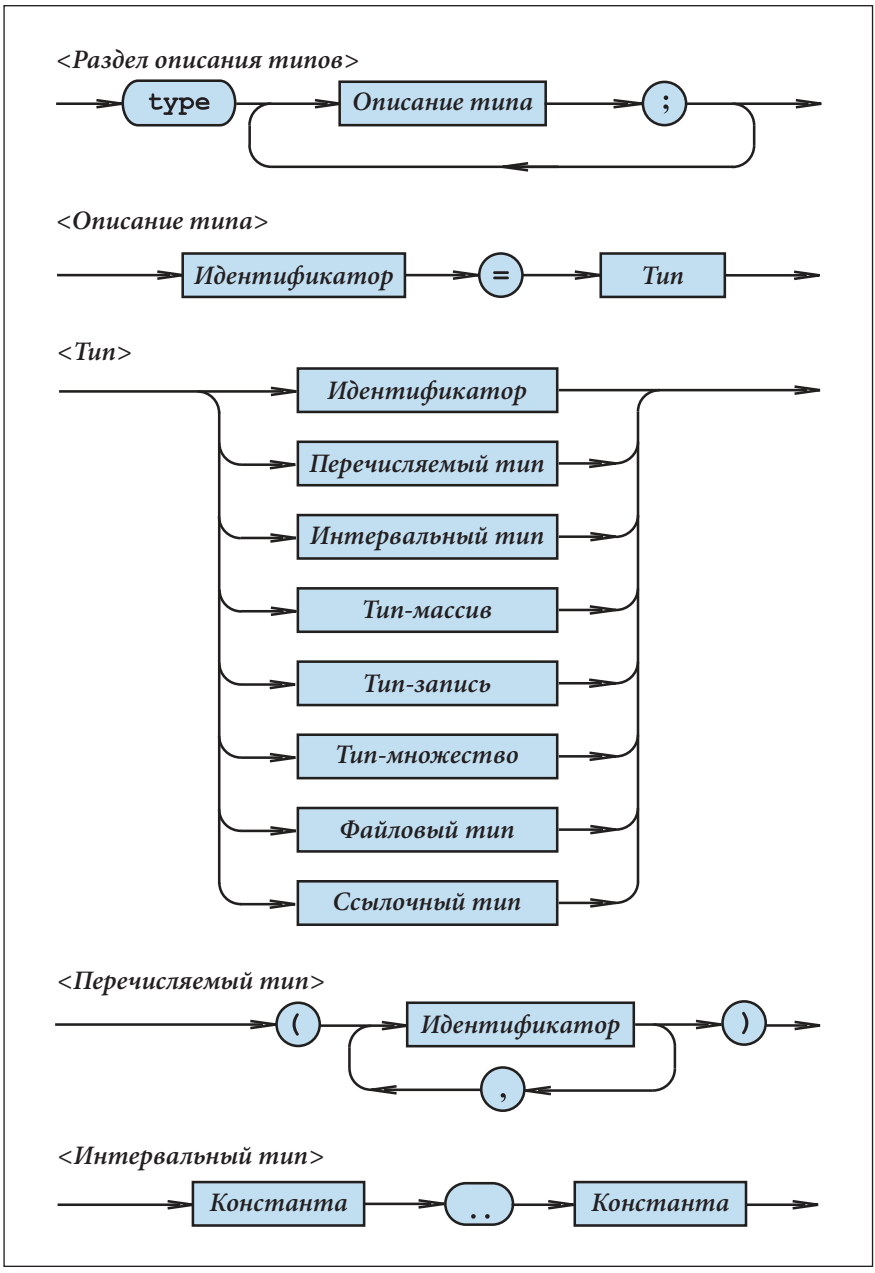


Рис. 2.2. Синтаксические диаграммы для описания типов данных

Например, пусть

```
type T4 = integer;
     T5 = integer;
```

Здесь типы `integer`, `T4`, `T5` являются идентичными.

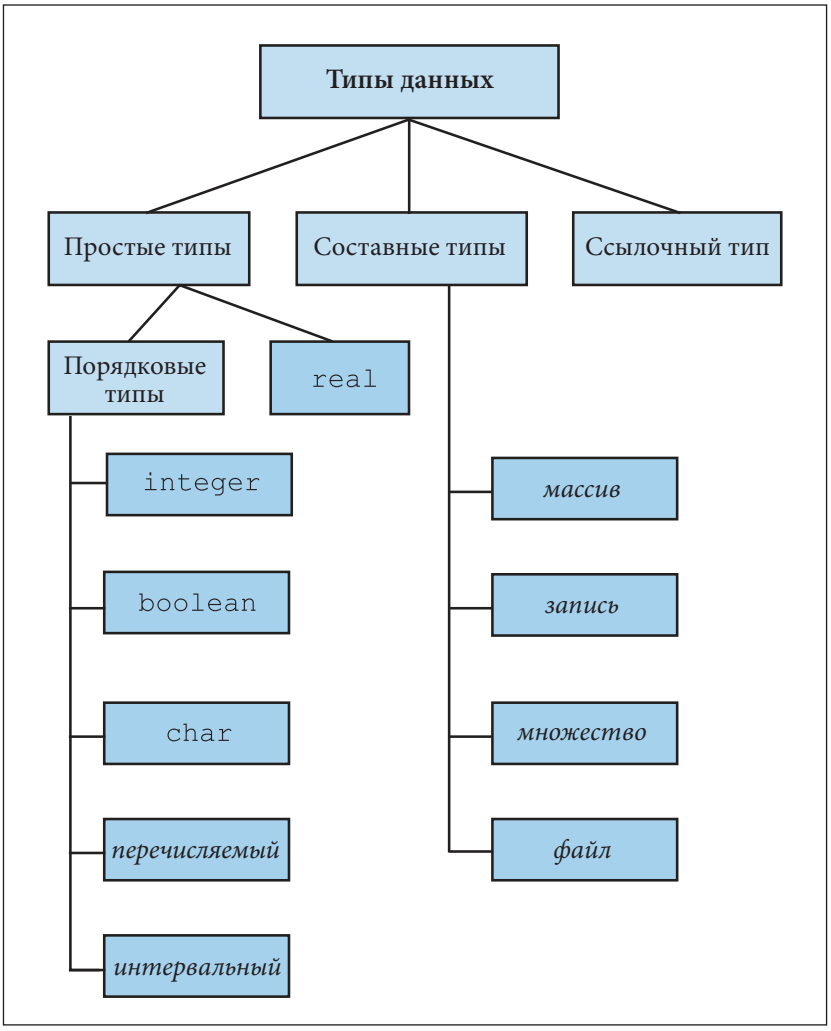


Рис. 2.3. Классификация типов данных

Два типа могут быть идентичными и тогда, когда они задаются различными именами, при условии, что эти имена эквивалентны по свойству транзитивности.

Например, пусть

```
type T6 = real;
     T7 = T6;
     T8 = T7;
```

Здесь типы `real`, `T6`, `T7` и `T8` являются идентичными.

Два типа называются **совместимыми** тогда, когда верно хотя бы одно из следующих утверждений:

1) рассматриваемые типы идентичны;
2) один тип является интервальным типом второго;
3) оба типа являются интервальными, с одним и тем же базовым типом.

Например, при следующем объявлении:

```
type Zi = (L, Ma, Mi, J, V, S, D);
     ZiDeLucru = (L, Ma, Mi, J, V);
     ZiDeOdihna = (S, D);
     Culoare = (Galben, Verde, Albastru, Violet);
```

типы `Zi`, `ZiDeLucru`, `ZiDeOdihna` являются совместимыми. Типы `Zi` и `Culoare` не совместимы. Отсюда следует, что допустимы следующие операции отношения:

```
L < D        Mi <> D        Verde <> Violet
```

и т.д., а операции типа

```
L < Violet   Verde = V      S <> Albastru
```

запрещены.

Кроме типов данных, определяемых пользователем явно с помощью ключевого слова **`type`**, в программе могут быть описаны анонимные типы (типы без имени).

Анонимный тип определяется неявно при описании переменных.

Пример:

```
var i : 1..20;
    s : (Alfa, Beta, Gama, Delta);
    t : Alfa..Gama;
```

Отметим, что интервальный тип `1..20`, *перечисляемый* тип (`Alfa, Beta, Gama, Delta`) и *интервальный* тип `Alfa..Gama` не имеют собственных имен.

Как правило, анонимные типы используются в программах с небольшим количеством переменных.

Вопросы и упражнения

❶ Дана следующая программа:

```
Program P28;
type  T1 = -100..100;
      T2 = 'A'..'H';
      T3 = (A, B, C, D, E, F, G, H);
      T4 = A..E;
      T5 = integer;
      T6 = real;
      T7 = char;
      T8 = boolean;
var   i : T1;
      j : T5;
      k : T2;
      m : T3;
      n : T4;
      p : real;
      q : T6;
      r : char;
```

```
      s : T7;
      t : boolean;
      y : real;
      z : T8;
begin
  { вычисления, в которых используются }
  { данные переменные }
  writeln('Конец');
end.
```

Укажите типы данных, используемые в программе. Какие значения может принимать каждая переменная, описанная в данной программе? Какие из указанных типов совместимы?

❷ Укажите на синтаксических диаграммах *рис.2.2* пути, соответствующие описаниям типов данных из программы P28.

❸ Укажите анонимные типы данных, используемые в следующей программе:

```
Program P29;
type  T1 = integer;
      T2 = -150..150;
      T3 = 1..5;
var   i : T1;
      j : T2;
      k : T3;
      m : 1..5;
      n : (Unu, Doi, Trei, Patru, Cinci);
begin
  { вычисления, в которых используются }
  { данные переменные }
  writeln('Конец')
end.
```

Какие значения может принимать каждая переменная, описанная в данной программе?

❹ Дано следующее объявление:

```
type  T1 = boolean;
      T2 = T1;
      T3 = T2;
      T4 = T3;
var x : T4;
```

Какие значения может принимать переменная x? Назовите операции соответствующего типа данных.

❺ Когда два типа данных являются идентичными? Приведите примеры.

❻ Когда два типа данных являются совместимыми? Приведите примеры.

❼ Дано объявление:

```
type  T1 = integer;
      T2 = T1;
      T3 = -5..+5;
      T4 = T3;
      T5 = -10..+10;
```

```
T6 = (A, B, C, D, E, F, G, H);
T7 = A..D;
T8 = E..H;
T9 = 'A'..'D';
T10 = 'E' ..'H';
```

Найдите идентичные и совместимые типы данных.

2.10. Описание переменных

Известно, что каждая встречающаяся в программе переменная должна быть сопоставлена с одним и только одним типом данных. Для этого используются следующие грамматические конструкции:

<Раздел описания переменных> ::= **var** *<Описание переменной>*; {*<Описание переменной>*;}

<Описание переменной> ::= *<Идентификатор>* {, *<Идентификатор>*} : *<Тип>*

Синтаксические диаграммы рассматриваемых грамматических единиц приведены на *рис. 2.4*.

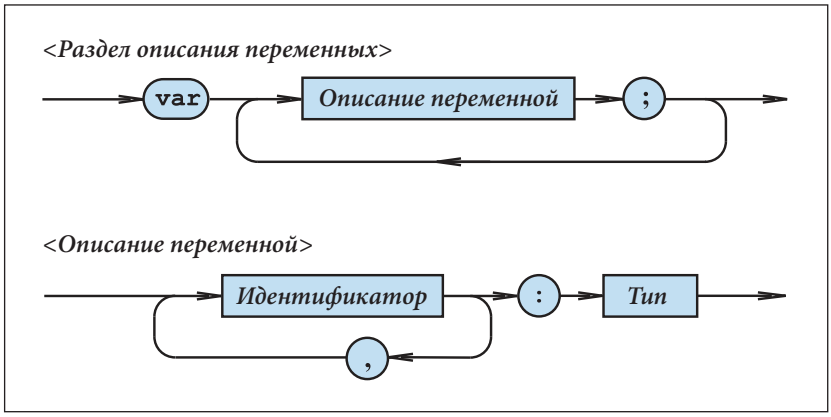


Рис. 2.4. Синтаксические диаграммы *<Раздел описания переменных>* и *<Описание переменной>*

В описании переменных могут использоваться предопределенные типы данных (`integer`, `real`, `char`, `boolean` и др.) и типы, определяемые пользователем (*перечисляемые, интервальные* и т.д.).

Примеры:

1)
```
var  i, j : integer;
     x : real;
     p : boolean;
     r, s : char;
```

2)
```
type T1 = (A, B, C, D, E, F, G, H);
     T2 = C..F;
     T3 = 1..10;
     T4 = 'A'..'Z';
var  x, y, z : real;
     r, s : char;
     i, j : integer;
     k : T3;
     p : T1;
     q : T2;
     v : T4;
```

3)
```
type Zi = (L, Ma, Mi, J, V, S, D);
var  x, y : real;
     z : Zi;
     zl : L..V;
     m, n : 1..10;
```

Отметим, что в последнем примере тип переменных `z1`, `m` и `n` описывается непосредственно в разделе описания переменных. Следовательно, эти переменные принадлежат анонимным типам данных.

Вопросы и упражнения

❶ Определите тип переменных, используемых в следующей программе:

```
Program P30;
type T1 = integer;
     Studii = (Elementare, Medii, Superioare);
     T2 = real;
     Culoare = (Galben, Verde, Albastru, Violet);
var  x : real;
     y : T1;
     i : integer;
     j : T2;
     p : boolean;
     c : Culoare;
     s : Studii;
     q : Galben..Albastru;
     r : 1..9;
begin
  { вычисления, в которых используются }
  { данные переменные }
  writeln('Конец');
end.
```

Какие значения может принимать каждая переменная? Назовите операции соответствующего типа данных.

❷ Укажите на синтаксических диаграммах *рис. 2.4* пути, которые соответствуют описанию переменных в программе P30.

❸ Какие сообщения будут выведены на экране в процессе компиляции следующей программы?

```
Program P31;
var i, j : integer;
begin
  i:=1; j:=2; k:=i+j;
  writeln('k=', k);
end.
```

❹ Как описываются переменные, принадлежащие анонимным типам данных?

2.11. Описание констант

Известно, что значения любого типа данных могут быть выражены через переменные или константы. Для того чтобы сделать программы более удобными для чтения и модификации, язык ПАСКАЛЬ позволяет представлять константы в виде символических имен. Идентификатор, представляющий константу, называется *именем константы* или просто *константой*. Везде, где в программе встречается такое имя, оно заменяется на соответствующее значение.

Константы описываются с помощью следующих грамматических конструкций:

<Раздел описания констант> ::= **const** *<Описание константы>*; { *<Описание константы>*;}

<Описание константы> ::= *<Идентификатор>* = *<Константа>*

<Константа> ::= [+ | -] *<Число без знака>* | [+ | -] *<Имя константы>* | *<Строка символов>*

Синтаксические диаграммы данных грамматических единиц приведены на *рис. 2.5.*

Примеры:

1)
```
const a = 10;
      b = 9.81;
      c = '*';
      t = 'TEXT';
```

2)
```
const NumarCaractere = 60;
      LungimePagina = 40;
```

3)
```
const n = 10;
      m = 20;
      Pmax = 2.15e+8;
      Pmin = -Pmax;
      S = 'STOP';
```

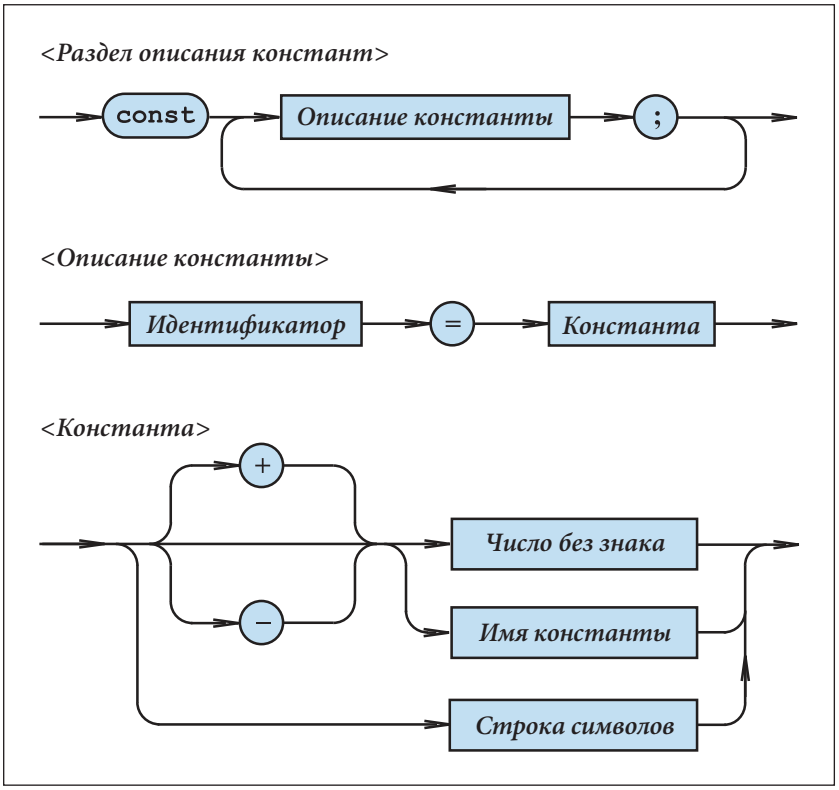


Рис. 2.5. Синтаксические диаграммы *<Раздел описания констант>*, *<Описание константы>* и *<Константа>*

В отличие от переменных, типы которых объявляются явно в соответствующих описаниях, типы констант объявляются неявно через их текстовую форму. В приведенных выше примерах типами констант являются:

`a, NumarCaractere, LungimePagina, n, m` – *integer;*
`b, Pmax, Pmin` – *real;*
`c` – *char;*
`t, S` – *строка символов.*

В программе `P32` описываются константы `Nmax`, `Nmin`, `Pi`, `Separator`, `Indicator` и `Mesaj`. Значения этих констант выводятся на экран.

```
Program P32;
 { Definitii de constante }
const Nmax = 40;          { Константа типа integer }
      Nmin = -Nmax;       { Константа типа integer }
      Pi = 3.14;          { Константа типа real }
      Separator = '\';    { Константа типа char }
      Indicator = TRUE;   { Константа типа boolean }
      Mesaj = 'Проверьте принтер'; { Строка символов }
```

```
begin
 writeln(Nmax);
 writeln(Nmin);
 writeln(Pi);
 writeln(Separator);
 writeln(Indicator);
 writeln(Mesaj);
 { вычисления, в которых используются }
 { данные константы }
end.
```

Обычно данные, которые не изменяются в процессе выполнения программы, например, количество строк таблицы, число π, ускорение свободного падения *g* и т.д., описываются в виде констант. Это позволяет изменять постоянные значения, не изменяя всей программы в целом.

В программе P33 длина *L* и площадь *S* круга вычисляются по формулам:

$$L=2\pi r;\ S=\pi r^2,$$

где *r* является радиусом окружности. Число π представлено в виде константы `Pi = 3.14`.

```
Program P33;
 { Длина и площадь круга }
const Pi = 3.14;
var L, S , r : real;
begin
 writeln('Введите радиус r:');
 readln(r);
 L:=2*Pi*r;
 writeln('Длина L=', L);
 S:=Pi*r*r;
 writeln('Площадь S=', S);
end.
```

Если пользователю необходимы более точные результаты, изменяется только третья строка программы P33:

```
const Pi = 3.141592654;
```

Оставшаяся часть программы остается неизменной.

В отличие от переменных, значения констант не могут изменяться посредством операции присваивания или считывания. Несоблюдение данного правила приводит к ошибкам, которые обнаруживаются при компиляции программы.

Вопросы и упражнения

❶ Определите типы данных следующих констант:

a)
```
const a = 29.1;
      b = TRUE;
      c = 18;
```

b)
```
const  d = -16.82e-14;
       f = -d;
       i = 15;
       j = '15';
       n = '-d';
```

c)
```
const  t = 'F';
       q = '1';
       c = '18';
```

d)
```
const  x = 65;
       q = FALSE;
       y = -x;
       m = 'PAUZĂ';
       z = -y;
```

❷ Напишите программу, которая выводит на экран значения констант из упражнения 1.

❸ Покажите на синтаксических диаграммах *рис. 2.5* пути, которые соответствуют описаниям констант из программы P32.

❹ Между ключевыми словами **begin** и **end** программы P32 вставьте одну из следующих строк:

a)
```
Nmax:=10;
```

b)
```
readln(Nmax);
```

Объясните сообщения, выводимые на экран в процессе компиляции модифицированной программы.

❺ Дана программа:

```
Program P34;
const t = '1';
      s = -t;
begin
     writeln(s);
end.
```

Какие сообщения будут выведены на экран в процессе компиляции?

❻ Что появится на экране после запуска на выполнение следующей программы:

```
Program P35;
const i = 1;
      j = i;
      k = j;
var x, y, z : integer;
begin
 x:=i+1; writeln(x);
 y:=j+2; writeln(y);
 z:=k+3; writeln(z);
end.
```

❼ Дана программа:

```
Program P36;
const a = 1;
      b = 2;
```

```
      c = 3;
      d = 4;
var i, j, k : integer;
begin
 i:=a+1; writeln(i);
 j:=b+1; writeln(j);
 k:=c+1; writeln(k);
 d:=a+1; writeln(d);
end.
```

Объясните сообщения, выводимые на экран.

❽ Исключите из следующей программы строку, в которой содержится ошибка.

```
Program P37;
const a = 1;
var i, j : integer;
begin
 writeln('Введите i=');
 readln(i); j:=i+a;
 writeln(j);
 writeln('Введите a=');
 readln(a);
 j:=i+a;
 writeln(j);
end.
```

Что появится на экране после запуска этой программы на выполнение?

Тест для самопроверки № 2

1. Объясните значение термина *тип данных*. Назовите, по крайней мере, два типа данных и приведите по несколько примеров данных соответствующих типов.

2. Каким образом представляются данные в программе на языке ПАСКАЛЬ? Объясните значения терминов *величина, переменная* и *константа*. Какова разница между переменной и константой?

3. Укажите типы данных используемые в следующей программе:

```
Program TA2;
 { Простые типы данных }
var  i, j : integer;
     a, b, c : real;
     s : char;
     p : boolean;
begin
   i:=5; j:=i+9;
   writeln(i); writeln(j);
   a:=1.0; b:=1.0e-01; c:=-2.001;
```

```
   writeln(a); writeln(b); writeln(c);
   s:='A'; writeln(s);
   p:=true; writeln(p);
end.
```

Пример: i – переменная типа integer; a – переменная типа real; 5 – константа типа integer и т.д.

4. Назовите множество значений типа данных integer? Какие операции можно выполнять с этими значениями?

5. Дана следующая программа на языке ПАСКАЛЬ:

```
Program TA3;
 { Ошибки переполнения }
var x, y, z : integer;
begin
   writeln('Введите целые числа x, y:');
   readln(x, y);
   z:=x*y; writeln('x*y=', z);
end.
```

Приведите примеры значений переменных x и y из программы TA3 при которых появляются ошибки переполнения.

6. Каково множество значений типа данных real? Какие операции допустимы с этими значениями? Являются ли результаты этих операций точными?

7. Дана следующая программа на языке ПАСКАЛЬ:

```
Program TA4;
 { Ошибки переполнения }
var x, y, z : real;
begin
   writeln('Введите вещественные числа x, y:');
   readln(x, y);
   z:=x*y; writeln('x*y=', z);
end.
```

Приведите примеры значений переменных x и y из программы TC4, при которых появляются ошибки переполнения.

8. Каково множество значений типа данных boolean? Какие операции допустимы с этими значениями?

9. Напишите таблицы истинности для логических операций **not**, **and** и **or**.

10. Укажите ошибку в следующей программе:

```
Program TA5;
 { Ошибка }
var p, q, r : boolean;
```

```
begin
  writeln('Введите логические значения p, q:');
  readln(p, q);
  r:=p and q;
  writeln(q);
end.
```

11. Каково множество значений типа данных `char`? Как упорядочено это множество? Какие операции можно выполнять со значениями типа `char`?

12. Напишите программу, которая выводит на экран порядковые номера десятичных цифр.

13. Каково множество значений *перечисляемого* типа данных? Как упорядочено это множество? Какие операции можно выполнять с этими значениями?

14. Напишите программу, которая выводит на экран порядковые номера значений следующих *перечисляемых* типов:

a)
```
FunctiaOcupata = (Muncitor, SefDeEchipa, Maistru,
                  SefDeSantier, Director);
```

b)
```
StareaCivila = (Casatorit, Necasatorit);
```

15. Каким образом определяется тип данных *интервал*? Каково множество значений типа данных *интервал*? Какие операции допустимы с этими значениями?

16. Какие значения может принимать каждая из объявленных ниже переменных?

```
type  T1 = 'A'..'Z';
      T2 = 1..9;
      T3 = '0'..'9';
      T4 = (Alfa, Beta, Gama, Delta);
var   p : T1;
      q : T2;
      r : T3;
      s : char;
      t : integer;
      u : T4;
```

Назовите базовый тип каждого из типов данных *интервал*.

17. Назовите порядковые типы данных. Каковы их общие свойства?

18. Что выводит на экран следующая программа?

```
Program TA6;
type Culoare = (Galben, Verde, Albastru, Violet);
begin
  writeln(pred('Z'));
  writeln(succ('D'));
  writeln(pred(-5));
  writeln(succ(9));
```

```
  writeln(ord(Verde));
  writeln(ord(Violet));
end.
```

19. Что выводит на экран следующая программа?

```
Program TA7;
type Nivel = (A, B, C, D, E, F, G);
var n, m : Nivel;
begin
  n:=B; writeln(ord(n));
  m:=pred(n); writeln(ord(m));
  m:=succ(n); writeln(ord(m));
  n:=C; m:=E;
  writeln(n<m);
  writeln(n>m);
  writeln(n<>m);
end.
```

20. Объясните следующие термины: идентичные типы, совместимые типы, анонимные типы.

21. Заданы объявления:

```
type  T1 = integer;
      T2 = T1;
      T3 = -5..+5;
      T4 = T3;
      T5 = -10..+10;
      T6 = (A, B, C, D, E, F, G, H);
      T7 = A..D;
      T8 = E..H;
      T9 = 'A'..'D';
      T10 = 'E' ..'H';
var   x : 1..100;
      y : (Alfa, Beta, Gama, Delta);
```

Укажите идентичные типы, совместимые типы и анонимные типы данных.

22. Определите тип каждой из следующих констант:

```
const  Alfa = 5;
       Beta = 12.485;
       Indicator = true;
       Mesaj = 'Eroare de executie';
       Semn = '+';
       Inscris = '12.485';
```

23. В некотором алгоритме используются целые переменные *i*, *j*, вещественные переменные *x*, *y*, константы 3,14 и 9,8. Напишите на языке ПАСКАЛЬ объявления для описания соответствующих переменных и констант.

Глава 3

ОПЕРАТОРЫ

3.1. Концепция действия

Согласно **концепции действия**, реализованой в языке ПАСКАЛЬ, компьютер является исполнителем, рабочая среда которого состоит из множества всех переменных и констант, объявленных в программе. В ходе выполнения программы исполнитель осуществляет над величинами из рабочей среды определенные действия (операции), например, сложение или вычитание, считывание с клавиатуры или вывод на экран и т.п. Очевидно, в результате этих действий значения переменных могут изменяться, в то время как значения констант – нет.

Действия, необходимые для обработки данных, и порядок их выполнения задаются с помощью **операторов**. Существуют две категории операторов:

1) простые операторы;
2) сложные операторы.

Простые операторы не содержат других операторов. Простыми операторами являются:

– оператор присваивания;
– оператор процедуры;
– оператор перехода;
– пустой оператор.

Сложные операторы составлены из других операторов. Сложными операторами являются:

– составной оператор **`begin...end`**;
– условные операторы **`if, case`**;
– операторы цикла **`for, while, repeat`**;
– оператор **`with`**.

Операторы **`if`** и **`case`** применяются для программирования алгоритмов с разветвлениями, а операторы **`for`**, **`while`** и **`repeat`** – для программирования циклических алгоритмов. Напомним, что циклические алгоритмы используются для описания многократно повторяющихся действий, а алгоритмы с разветвлениями – для выбора необходимых действий в зависимости от условий из рабочей среды исполнителя.

Каждому оператору может предшествовать метка. Ссылка на метку указывается в операторе перехода. Напомним, что метка – это целое число без знака (см. параграф 1.5.5).

Синтаксическая диаграмма <*Оператор*> показана на *рис. 3.1.* Напоминаем, что метка отделяется от оператора с помощью символа «**`:`**» (двоеточие).

В программах на языке ПАСКАЛЬ количество операторов в строке не ограничено – один оператор может занимать одну или более строк, а в одной строке может

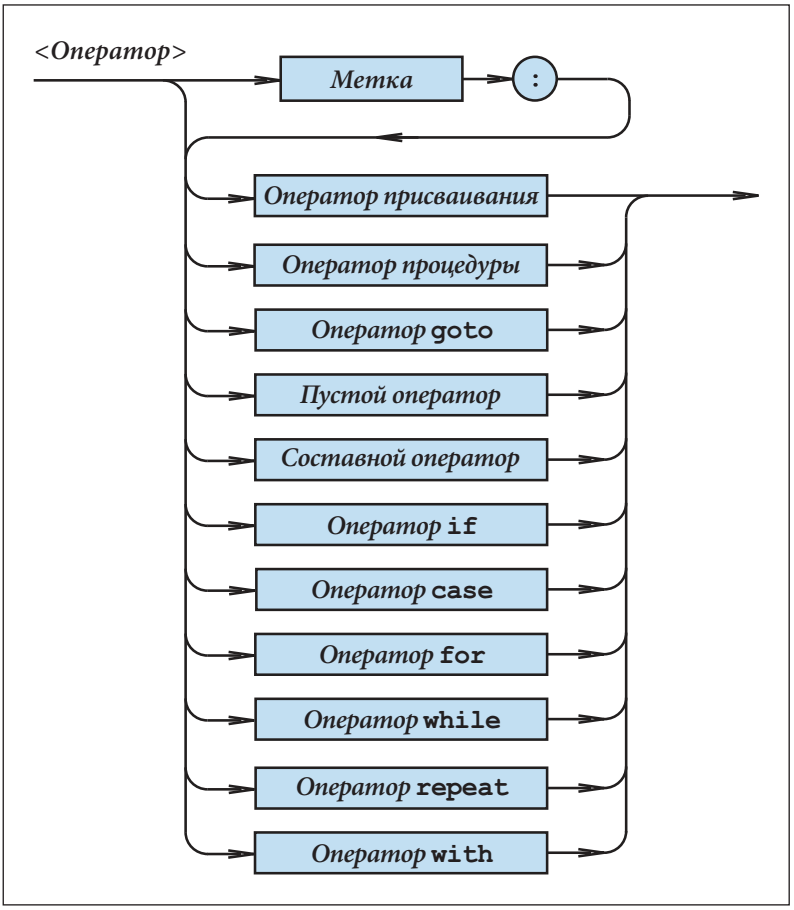


Рис. 3.1. Синтаксическая диаграмма <*Оператор*>

быть несколько операторов. В качестве разделителя операторов используют символ «**;**» (точка с запятой).

3.2. Выражения

Формулы для вычисления значений представляются на языке ПАСКАЛЬ в виде **выражений**. Выражения состоят из операндов (констант, переменных, ссылок на функции) и операций. Операции классифицируются следующим образом:

<*Мультипликативная операция*> ::= ***** | **/** | **div** | **mod** | **and**

<*Аддитивная операция*> ::= **+** | **–** | **or**

<*Операция отношения*> ::= **<** | **<=** | **=** | **>=** | **>** | **<>** | **in**

Выражения состоят из факторов (множителей), термов (слагаемых) и простых выражений.

Фактор состоит из отдельных переменных, констант без знака, обозначений функций и др.

<Фактор> ::= *<Переменная>* | *<Константа без знака>* | *<Вызов функции>* | **not** *<Фактор>* | (*<Выражение>*) | *<Конструктор множества>*

Примеры:

1) `15`

2) `x`

3) `p`

4) `sin(x)`

5) `not p`

6) `cos(x)`

Терм имеет вид:

<Терм> ::= *<Фактор>* {*<Мультипликативная операция>* *<Фактор>*}

Примеры:

1) `15`

2) `x`

3) `15*x`

4) `x*y*z`

5) `p and q`

6) `sin(x)/3`

Под **простым выражением** понимается:

<Простое выражение> ::= `[+|-]`*<Терм>*{*<Аддитивная операция>* *<Терм>*}

Примеры:

1) `15`

2) `+15`

3) `15*x+sin(x)/3`

4) `p or q`

5) `-a+b-c`

6) `sin(x)/3+cos(x)`

В свою очередь **выражение** имеет вид:

<Выражение> :: = *<Простое выражение>* {*<Операция отношения>* *<Простое выражение>*}

Примеры:

1) `-15`

2) `x*y+cos(z)/4`

3) `x<15`

4) `15*x+sin(x)/3<>11`

5) `a>c`

6) `x+6>z-3.0`

Синтаксические диаграммы рассмотренных грамматических единиц приведены на *рис. 3.2* и *3.3*.

Выражение, взятое в скобки, преобразуется в фактор. Факторы могут образовывать новые термы, простые выражения, выражения.

Примеры:

	Обычная форма записи	*Запись на языке ПАСКАЛЬ*
1)	$\frac{a+b}{c+d}$	`(a+b)/(c+d)`

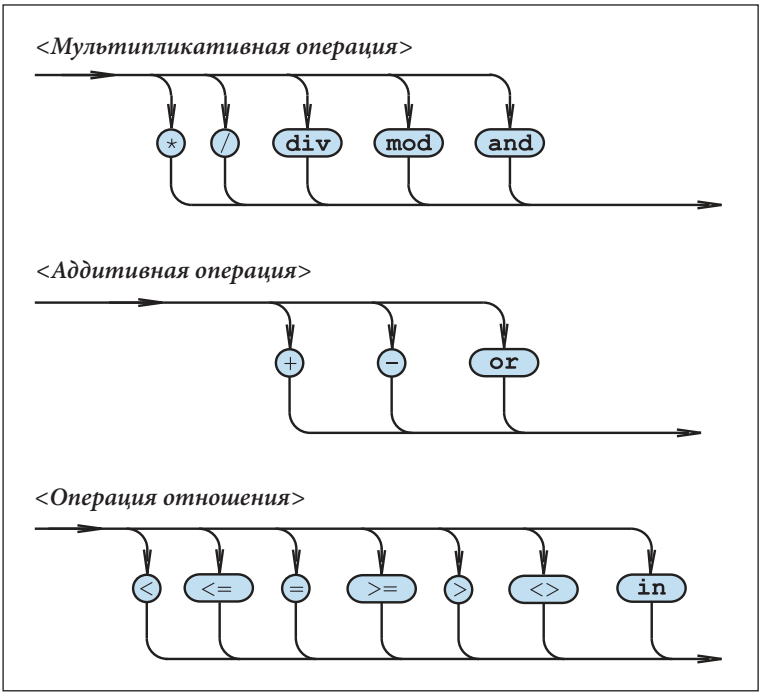


Рис. 3.2. Синтаксические диаграммы операций

2)	$\frac{-b+\sin(b-c)}{2a}$	`(-b+sin(b-c))/(2*a)`
3)	$-\frac{1}{xy}$	`-1/(x*y)`
4)	$p < q \,\&\, r > s$	`(p<q)and(r>s)`
5)	$\overline{x \vee y}$	`not (x or y)`
6)	$\frac{1}{a+b} > \frac{1}{c+d}$	`1/(a+b)>1/(c+d)`

Вызов функции может появляться везде, где допустимо присутствие константы или переменной.

Язык ПАСКАЛЬ содержит ряд **стандартных функций**, известных любой программе. Эти функции представлены в *таблице 3.1.*

Вопросы и упражнения

❶ Перепишите следующие выражения согласно правилам языка ПАСКАЛЬ:

a) a^2+b^2; *b)* $a^2 + 2ab + b^2$;

Таблица 3.1.

Стандартные функции языка ПАСКАЛЬ

Название функции	Обозначение в языке ПАСКАЛЬ
Абсолютное значение $\|x\|$	`abs(x)`
Синус sin x	`sin(x)`
Косинус cos x	`cos(x)`
Арктангенс arctg x	`arctan(x)`
Квадрат x^2	`sqr(x)`
Квадратный корень $\sqrt{x}$	`sqrt(x)`
Экспоненциальная функция e^x	`exp(x)`
Натуральный логарифм ln x	`ln(x)`
Округление вещественного числа x	`round(x)`
Целая часть вещественного числа x	`trunc(x)`
Проверка числа i на нечетность (возвращает значение `true`, если i – нечетно и `false` в противном случае)	`odd(i)`
Порядковый номер элемента v	`ord(v)`
Предшествующий элемент v	`pred(v)`
Следующий элемент v	`succ(v)`
Символ с порядковым номером i	`chr(i)`
Признак конца файла	`eof(f)`
Признак конца строки	`eoln(f)`

c) $(a + b)^2$;

d) $v_0 t + \frac{at^2}{2}$;

e) $\frac{-b + \sqrt{b^2 - 4ac}}{2a}$;

f) $\cos \alpha + \cos \beta$;

g) $\cos (\alpha + \beta)$;

h) $2\pi r$;

i) πr^2;

j) $x_1 x_2 \vee x_3 x_4$;

k) $\overline{x_1 \vee x_2}$;

l) $|x| < 3$;

m) $|z| < 6 \;\&\; |q| > 3{,}14$;

n) $x > 0 \;\&\; y > 8 \;\&\; R < 15$.

❷ Укажите на синтаксических диаграммах *рис. 3.3* пути, которые соответствуют следующим выражениям:

a) `x`

b) `3.14`

c) `sin(x)`

d) `not q`

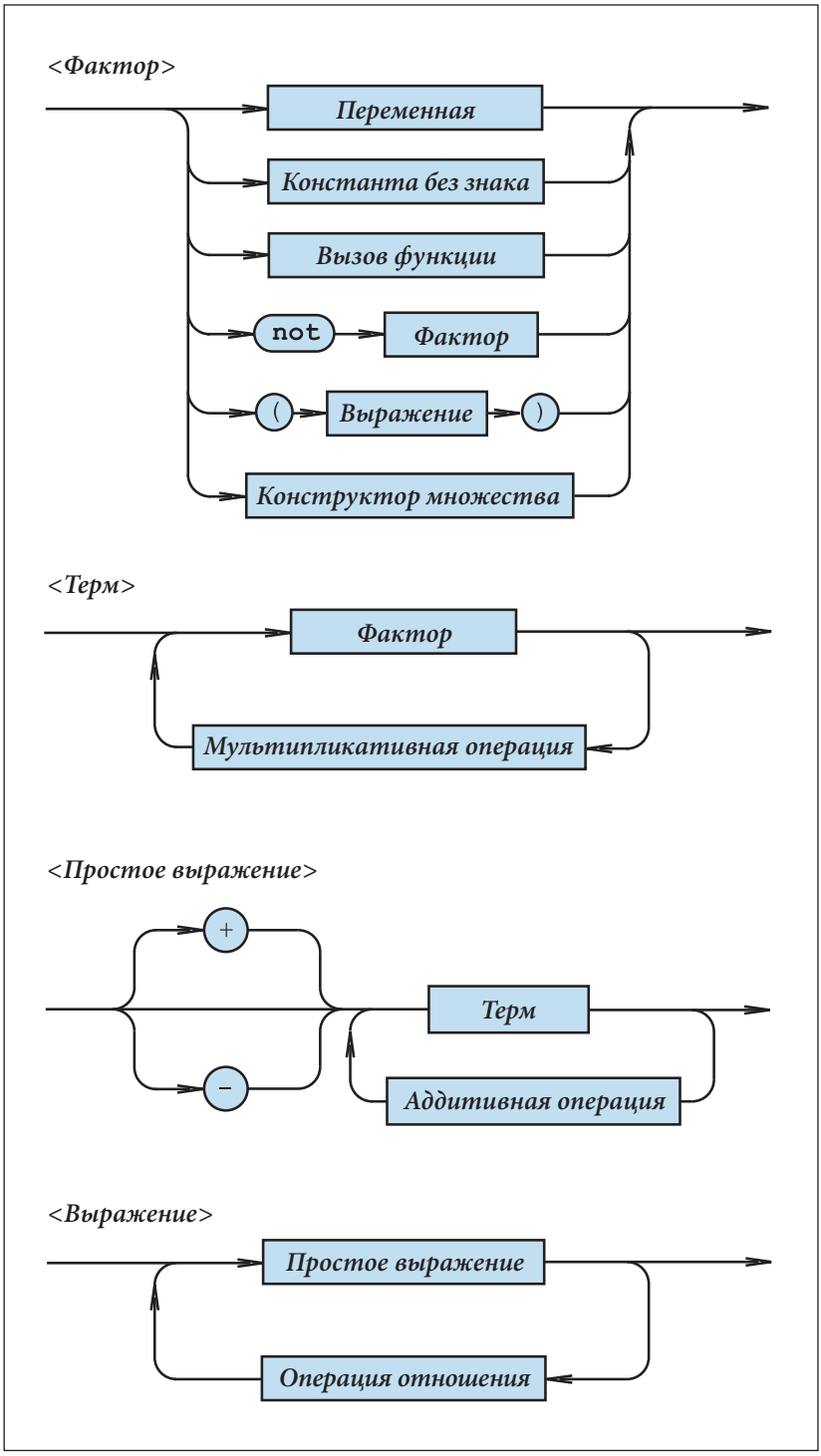


Рис. 3.3. Синтаксические диаграммы для определения выражений

e) `(+3.14)`

f) `x>2.85`

g) `sqr(b)-4*a*c>0`

h) `(a>b)and(c>d)`

❸ Переведите выражения, записанные на языке ПАСКАЛЬ, в обычные:

a) `sqr(a)+sqr(b)`

b) `2*a*(b+c)`

c) `sqrt((a+b)/(a-b))`

d) `exp(x+y)`

e) `cos(ALFA-BETA)`

f) `sqr(a+b)/(a-b)`

g) `x>0 or q<p`

h) `not(x and y)`

❹ Какие из следующих выражений на языке ПАСКАЛЬ содержат ошибки? Для ответа воспользуйтесь синтаксическими диаграммами *рис. 3.3*.

a) `((((+x))))`

b) `((((x))))`

c) `sinx+cosx`

d) `sqr(x)+sqr(y)`

e) `a<<b or c>d`

f) `not not not p`

g) `q and not p`

h) `not q and p`

i) `a+-b`

j) `sin(-x)`

k) `sin-x`

l) `cos(x+y)`

m) `sin(abs(x)+abs(y))`

n) `sqrt(-y)`

3.3. Вычисление выражений

Под вычислением некоторого выражения понимается нахождение его значения. Результат вычисления зависит от значений операндов и от операций, применяемых к ним. Правила вычисления выражений – такие же, как и в математике:

– операции выполняются в соответствии с их приоритетом;

– в случае одинаковых приоритетов операции выполняются слева направо;

– первыми вычисляются значения выражений, заключенных в скобках.

Приоритеты операций приведены в *таблице 3.2*.

Таблица 3.2.

Приоритеты операций языка ПАСКАЛЬ

Категория	Операция	Приоритет
Унарные операции	**not**, @	Первый (наивысший)
Мультипликативные операции	*, /, **div**, **mod**, **and**	Второй
Аддитивные операции	+, -, **or**	Третий
Операции отношения	<, <=, =, >=, >, <>, **in**	Четвертый (наименьший)

Пример:

Пусть x=2 и y=6. Тогда:

1) `2*x+y = 2*2+6 = 4+6 = 10;`

2) `2*(x+y) = 2(2+6) = 2·8 = 16;`

3) `x+y/x–y = 2+6/2–6 = 2+3–6 = 5–6 = –1;`

4) `(x+y)/x–y = (2+6)/2–6 = 8/2–6 = 4–6 = –2;`

5) `x+y/(x–y) = 2+6/(2–6) = 2+6/(–4) = 2+(–1,5) = 0,5;`

6) `x+y<15 = 2+6<15 = 8<15 = true;`

7)
```
(x+y<15)and(x>3) = (2+6<15)and(2>3) = (8<15)and(2>3) =
                 = true and false = false.
```

Отметим, что составные части выражения (*рис. 3.3*) вычисляются в следующем порядке:

1) факторы;
3) простые выражения;
2) термы;
4) само выражение.

Текущее значение выражения можно вывести на экран с помощью процедуры `writeln`:

`writeln(`*<Выражение>*`)`

Программа P38 выводит на экран результаты вычисления выражений `x*y+z` и `x+y<z–1.0`. Текущие значения переменных x, y и z считываются с клавиатуры.

```
Program P38;
 { Вычисление выражений }
var x, y, z : real;
begin
 writeln('Введите вещественные числа x, y, z:');
 readln(x, y, z);
 writeln(x*y+z);
 writeln(x+y<z-1.0);
end.
```

Вопросы и упражнения

❶ Пусть x=1, y=2 и z=3. Вычислите следующие выражения:

a) `x+y+2*z`

b) `(x+y+2)*z`

c) `x*y+y*z`

d) `x*(y+y)*z`

e) `(x*y+y)*z`

f) `x*(y+y*z)`

g) `x*y<y*z`

h) `(x>y)or(6*x>y+z)`

i) `not(x+y+z>0)`

j) `not(x+y>0) and not(z<0)`

❷ Сформулируйте правила вычисления выражений на языке ПАСКАЛЬ.

❸ Назовите приоритеты операций языка ПАСКАЛЬ.

❹ Укажите порядок вычисления выражений.

❺ Напишите программу, которая вычисляет выражения *c)* и *g)* из упражнения 1. Текущие значения вещественных переменных `x`, `y` и `z` считываются с клавиатуры.

3.4. Тип выражений

Каждому выражению, в зависимости от множества его значений, ставится в соответствие определенный тип данных. Согласно концепции данных, реализованной в языке ПАСКАЛЬ, **тип выражения** определяется типами операндов и операциями, применяемыми к ним. Таким образом, тип выражения можно определить не вычисляя его значения.

Типы результатов операций указаны в *таблице 3.3*. *Таблица 3.4* содержит тип результатов стандартных функций языка ПАСКАЛЬ.

Независимо от типа операндов операция `/` (деление) возвращает результат типа `real`, а операции отношения возвращают результат типа `boolean`.

Для того чтобы определить тип выражения, факторы, термы и простые выражения рассматриваются в порядке их вычисления, причем тип каждой составной части определяется с помощью *таблиц 3.3* и *3.4*.

Например, рассмотрим выражение:

```
(x>i) or (6*i<sin(x/y))
```

в котором `i` – переменная типа `integer`, а `x` и `y` – переменные типа `real`.

Таблица 3.3.

Типы результатов операций

Операция	Тип операндов	Тип результата
+, -, *	`integer`	`integer`
	Один `integer`, другой `real`	`real`
`/`	`integer` или `real`	`real`
`div`	`integer`	`integer`
`mod`	`integer`	`integer`
`not`, **`and`**, **`or`**	`boolean`	`boolean`
<, <=, =, >=, >, <>	Идентичные типы	`boolean`
	Совместимые типы	`boolean`
	Один `integer`, другой `real`	`boolean`

Таблица 3.4.

Типы результатов стандартных функций

Функция	Тип аргумента	Тип результата
`abs(x)`	`integer` или `real`	Совпадает с типом `x`
`sin(x)`	`integer` или `real`	`real`
`cos(x)`	`integer` или `real`	`real`
`arctan(x)`	`integer` или `real`	`real`
`sqr(x)`	`integer` или `real`	Совпадает с типом `x`
`sqrt(x)`	`integer` или `real`	`real`
`exp(x)`	`integer` или `real`	`real`
`ln(x)`	`integer` или `real`	`real`
`round(x)`	`real`	`integer`
`trunc(x)`	`real`	`integer`
`odd(i)`	`integer`	`boolean`
`ord(v)`	Порядковый	`integer`
`pred(v)`	Порядковый	Совпадает с типом `v`
`succ(v)`	Порядковый	Совпадает с типом `v`
`chr(i)`	`integer`	`char`
`eof(f)`	Файловый	`boolean`
`eoln(f)`	Файловый	`boolean`

Определим тип составных частей и всего выражения в порядке выполнения вычислений:

1) `x>I` — `boolean`;

2) `6*i` — `integer`;

3) `x/y` — `real`;

4) `sin(x/y)` — `real`;

5) `6*i < sin(x/y)` — `boolean`;

6) `(x>i)or(6*i<sin(x/y))` — `boolean`.

Таким образом, рассматриваемое выражение относится к типу `boolean`.

Так как в процессе определения типов конкретные значения соответствующих выражений не вычисляются, интервальные типы расширяются до базовых.

Например, при следующих описаниях:

```
type T1=1..10;  { интервал типа integer }
     T2=11..20; { интервал типа integer }
```

```
var i:T1;
    j:T2;
```

имеем:

	Выражение	*Тип выражения*
1)	`i+j`	`integer`;
2)	`i mod j`	`integer`;
3)	`i/j`	`real`;
4)	`sin(i+j)`	`real`;
5)	`i>j`	`boolean`.

и т.д.

В зависимости от типа выражения разделяются на:
– арифметические (`integer` или `real`);
– порядковые (`integer`, `boolean`, `char`, перечисляемые);
– логические (`boolean`).

Как правило, арифметические выражения используются в вычислениях (оператор присваивания), порядковые выражения – в операторах **`case`** и **`for`**, а логические – в операторах **`if`**, **`repeat`** и **`while`**.

Вопросы и упражнения

❶ Как определяется тип выражения?

❷ При наличии описаний:

```
var x, y : real;
    i, j : integer;
    p, q : boolean;
    r : char;
    s : (A, B, C, D, E, F, G, H);
```

определите типы следующих выражений:

a)	`i mod 3`	*i)*	`sqr(i)-sqr(j)`
b)	`i/3`	*j)*	`sqr(x)-sqr(y)`
c)	`i mod 3 > j div 4`	*k)*	`trunc(x)+trunc(y)`
d)	`x+y/(x-y)`	*l)*	`chr(i)`
e)	`not(x<i)`	*m)*	`ord(r)`
f)	`sin(abs(i)+abs(j))`	*n)*	`ord(s)>ord(r)`
g)	`sin(abs(x)+abs(y))`	*o)*	`pred(E)`
h)	`p and (cos(x)<=sin(y))`	*p)*	`(-x+sin(x-y))/(2*i)`

❸ Тип выражения можно узнать из текстовой формы результатов, выведенных на экран с помощью оператора `writeln(`<*Выражение*>`)`.

Примеры:

	Результат, выведенный на экран	*Тип выражения*
1)	`100`	`integer;`
2)	`1.0000000000E+02`	`real;`
3)	`true`	`boolean.`

Напишите соответствующие программы и на основании текстовой формы результатов, выведенных на экран, определите типы следующих выражений:

a) `1+1.0`

b) `1/1+1`

c) `9*3 mod 4`

d) `4*x>9*y`

e) `chr(65)`

f) `not(x>y)`

g) `pred(9)>succ(7)`

h) `15 div ord(3)`

i) `trunc(x)+round(6*y)`

j) `sqr(3)-sqrt(16)`

где переменные `x` и `y` являются переменными типа `real` (вещественный).

❹ Даны описания:

```
type T1=1..10;
     T2=11..20;
     T3='A'..'Z';
     T4=(A, B, C, D, E, F, G, H);
var  i : T1;
     j : T2;
     k : T3;
     m : 'C'..'G';
     n : T4;
     p : C..G;
     q : boolean;
```

Найдите типы следующих выражений:

a) `i-j`

b) `i div j`

c) `6.3*i;`

d) `cos(3*i-6*j)`

e) `4*i>5*j`

f) `k<m`

g) `k<>m`

h) `chr(i)`

i) `ord(k)`

j) `ord(m)`

k) `n>p`

l) `ord(n)`

m) `succ(n)`

n) `pred(p)`

o) `ord(p)`

p) `ord(k)>ord(m)`

q) `(i>j) and q`

r) `not(i+j>0) or q`

3.5. Оператор присваивания

Оператор присваивания имеет вид:

<Переменная> `:=` *<Выражение>*

При выполнении оператора присваивания происходит следующее:
а) вычисляется выражение, стоящее в правой части;
б) полученное значение присваивается переменной, стоящей в левой части.

Примеры:

1) `x:=1`

2) `y:=x+3`

3) `z:=sin(x)+cos(y)`

4) `p:=not q`

5) `q:=(a<b) or (x<y)`

6) `c:='A'`

Отметим, что символ “`:=`” (читается «присвоить») означает присваивание и не следует путать его с операцией отношения “=” (равно).

Присваивание возможно только тогда, когда переменная и результат вычисления выражения **совместимы с точки зрения присваивания**. В противном случае возникает ошибка.

Переменная и результат вычисления выражения являются совместимыми с точки зрения присваивания, если справедливо одно из следующих утверждений:

1) тип переменной и тип результата идентичны;
2) тип результата является интервалом типа переменной;
3) оба типа являются интервалами одного и того же типа, а тип результата принадлежит интервальному типу переменной;
4) тип переменной – `real`, а тип результата – `integer` или его интервал.

В качестве примера рассмотрим следующую программу:

```
Program P39;
 { Совместимость с точки зрения присваивания }
type T1=1..10; { интервал типа integer }
     T2=5..15; { интервал типа integer }
var i : T1;
    j : T2;
    k, m, n : integer;
    x : real;
begin
 write('k=');
 readln(k);
 i:=k; { верно для 1<=k<=10 }
 write('m=');
 readln(m);
 j:=m; { верно для 5<=m<=15 }
 write('n=');
 readln(n);
```

```
 i:=n+5; { верно для -4<=n<=5 }
 j:=n+2; { верно для 3<=n<=13 }
 x:=i+j;
 writeln('x=', x);
end.
```

Программа P39 будет работать без ошибок только при следующих значениях на входе:

$$1 \le k \le 10; 5 \le m \le 15; 3 \le n \le 5.$$

Очевидно, что в программе P39 операторы вида

```
k:=x        i:=x+1        j:=sin(i)
```

являются ошибочными, так как тип выражений x, x+1, sin(i) – real, а тип переменных k, i, j – integer или его интервал.

Вопросы и упражнения

❶ Как выполняется оператор присваивания?

❷ Объясните термин «совместимость с точки зрения присваивания».

❸ Даны следующие описания:

```
type Zi = (L, Ma, Mi, J, V, S, D);
     Culoare = (Galben, Verde, Albastru, Violet);
var i, j, k : integer;
    z : Zi;
    c : Culoare;
    x : real;
```

Какие из следующих выражений являются правильными?

a) `i:=12`

b) `j:=ord(i)`

c) `x:=ord(z)+1`

d) `k:=ord(x)+2`

e) `c:=i+4`

f) `c:=Verde`

g) `z:=D`

h) `c:=Pred(Galben)`

i) `x:=Succ(z)`

j) `i:=Succ(c)`

❹ При каких значениях переменной j программа P40 будет работать без ошибок?

```
Program P40;
var i : -10..+10;
    j : integer;
begin
 write('j=');
 readln(j);
 i:=j+15;
 writeln('i=', i);
end.
```

3.6. Оператор процедуры

Процедура представляет собой подпрограмму, к которой в процессе выполнения программы можно обращаться произвольное число раз. Каждая процедура имеет свое имя, например: `readln`, `writeln`, `CitireDate`, `A15` и т. д. Язык ПАСКАЛЬ содержит ряд стандартных процедур, известных любой программе: `read`, `readln`, `write`, `writeln`, `get`, `put`, `new` и т. д. В дополнение к ним программист может создавать собственные процедуры.

Оператор процедуры *вызывает процедуру* с соответствующим именем. Оператор имеет вид:

<Оператор процедуры> ::= *<Имя процедуры>* [*<Список фактических параметров>*]

<Имя процедуры> ::= *<Идентификатор>*

<Список фактических параметров> ::= (*<Фактический параметр>*
{, *<Фактический параметр>*})

Синтаксические диаграммы указанных металингвистических формул представлены на *рис. 3.4*.

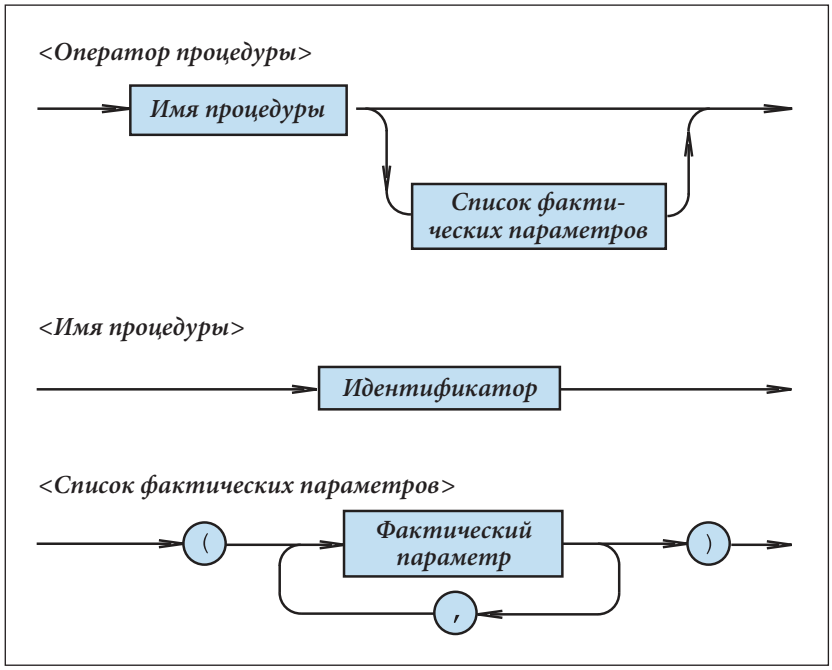


Рис. 3.4. Синтаксические диаграммы оператора процедуры

Как правило, *<Фактический параметр>* является выражением.

Примеры:

1) `readln(x)`

2) `readln(x, y, z)`

3) `CitireDate(f, t)`

4) `Exit`

5) `writeln(x+y, sin(x))` *6)* `writeln(2*x)`

Тип каждого фактического параметра и порядок его появления в списке указываются в разделе описаний соответствующих процедур. Правила составления списка фактических параметров будут изучены в следующих главах.

Вопросы и упражнения

❶ Для чего необходим оператор процедуры?

❷ Даны следующие операторы:

a) `readln(x, y, z, q)`

b) `CitireDate(ff, tt)`

c) `Halt`

d) `writeln('x=', x, 'y=', y)`

e) `writeln('x+y=', x+y, 'sin(x)=', sin(x))`

Укажите имена вызываемых процедур, число фактических параметров каждой процедуры, а также назовите эти фактические параметры.

❸ Укажите на синтаксических диаграммах *рис. 3.4* пути, которые соответствуют операторам из упражнения 2.

3.7. Вывод алфавитно-цифровой информации на экран

В большинстве версий языка ПАСКАЛЬ экран монитора является стандартным устройством вывода. Как правило, экран разделяется на условные зоны, называемые символьными. Указанные зоны образуют 25 строк по 80 символов в каждой. Зона, в которой будет выведен текущий символ, обозначается курсором.

Данные выводятся на экран с помощью процедуры `write(x)` или `writeln(x)`.

Оператор

```
write(x1, x2, ..., xn)
```

эквивалентен последовательности

```
write(x1); write(x2); ...; write(xn).
```

Фактические параметры процедур `write` или `writeln` называются **параметрами вывода**. Они могут иметь следующий вид:

e $\qquad$ $e : w$ $\qquad$ $e : w : f$

где e – выражение типа `integer`, `real`, `boolean`, `char` или строкового типа, значение которого нужно вывести на экран; w и f являются выражениями типа `integer` и называются **параметрами размера поля**. Значение выражения w указывает минимальное число символов, используемых для вывода значения e; если для

представления значения *e* требуется меньше, чем *w* символов, то ему предшествует такое число пробелов, чтобы было записано точно *w* символов (*рис. 3.5*).

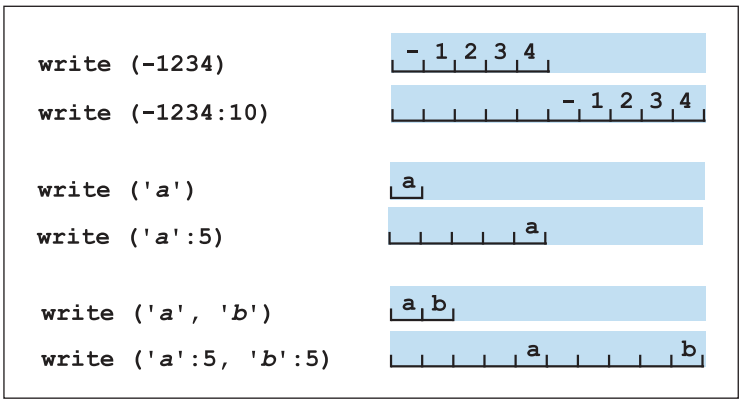


Рис. 3.5. Значение параметра размер поля *w*

Параметр размера поля *f* может присутствовать только в том случае, когда *e* является выражением типа `real`. Данный параметр указывает количество цифр после запятой в представлении значения выражения *e* с фиксированной точкой, без масштабного множителя. При отсутствии *f* значение *e* записывается с плавающей точкой, с масштабным множителем (*рис. 3.6*).

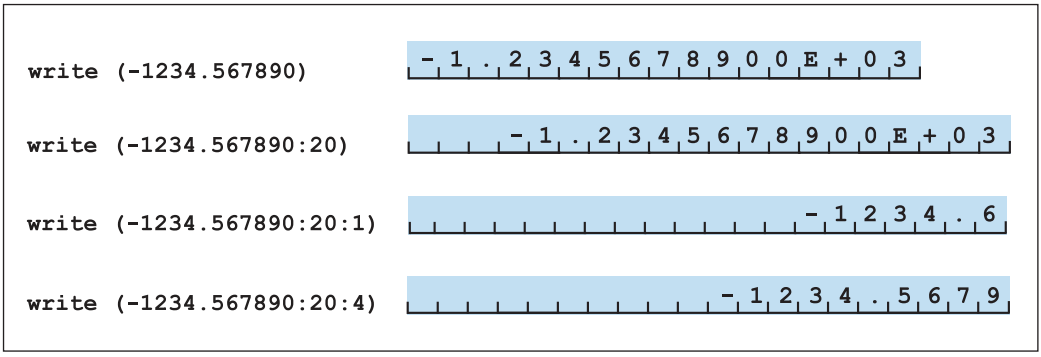


Рис. 3.6. Значение параметра размер поля *f*

Разница между процедурами `write` и `writeln` состоит в том, что после вывода данных `write` оставляет курсор в текущей строке, тогда как `writeln` переводит курсор на начало следующей строки. Рациональное использование процедур `write`, `writeln` и параметров размера поля обеспечивает вывод данных в форме, удобной для чтения. При выводе на экран нескольких значений рекомендуется указывать соответствующие им идентификаторы или сопровождать их комментариями.

Примеры:

1) `write('Сумма введенных чисел')`

2) `writeln(s:20)`

3) `writeln('Сумма=', s)`

4) `writeln('s=', s)`

5) `writeln('x=', x, 'y=':5, y, 'z=':5, z)`

Вопросы и упражнения

❶ Для чего нужен параметр *размер поля*?

❷ Как называются фактические параметры процедур `write` и `writeln`?

❸ Определите форматы данных, выводимых на экран следующими программами:

```
Program P41;
 { Вывод данных типа integer }
var i : integer;
begin
 i:=-1234;
 writeln(i);
 writeln(i:1);
 writeln(i:8);
 writeln(i, i);
 writeln(i:8, i:8);
 writeln(i, i, i);
 writeln(i:8, i:8, i:8);
end.
```

```
Program P42;
 { Вывод данных типа real }
var x : real;
begin
 x:=-1234.567890;
 writeln(x);
 writeln(x:20);
 writeln(x:20:1);
 writeln(x:20:2);
 writeln(x:20:4);
 writeln(x, x, x);
 writeln(x:20, x:20, x:20);
 writeln(x:20:4, x:20:4, x:20:4);
end.
```

```
Program P43;
 { Вывод данных типа boolean }
var p : boolean;
begin
 p:=false;
 writeln(p);
```

```
 writeln(p:10);
 writeln(p, p);
 writeln(p:10, p:10);
end.
```

```
Program P44;
 { Вывод символьных строк }
begin
 writeln('abc');
 writeln('abc':10);
 writeln('abc', 'abc');
 writeln('abc':10, 'abc':10);
end.
```

❹ Напишите программу, которая выводит на экран значения `1234567890`, `123`, `123.0` и `true` следующим образом:

```
1234567890
123
123.0
true
1234567890
                    123
                    123.000
                    true
```

3.8. Ввод данных с клавиатуры

Как правило, **стандартным устройством** ввода является клавиатура. Ввод данных с клавиатуры выполняется с помощью стандартных процедур `read` или `readln`. Список фактических параметров процедуры `read` или `readln` может включать переменные типа `integer`, `real`, `char` и *строкового типа*.

Оператор процедуры

```
read(x)
```

выполняет следующие действия: если переменная `x` является переменной типа `integer` или `real`, тогда считывается вся строка символов, представляющая целое или вещественное значение; если переменная `x` является переменной типа `char`, то процедура считывает только один символ.

Оператор процедуры

```
read(x1, x2, ..., xn)
```

эквивалентен последовательности

```
read(x1); read(x2); ...; read(xn).
```

Числа, вводимые с клавиатуры, должны разделяться пробелами или символами конца строки. Пробелы, стоящие перед самим числом, игнорируются. Строка симво-

лов, представляющая собой число, должна соответствовать синтаксису числовых констант соответствующего типа. В противном случае возникает ошибка ввода-вывода.

Например, рассмотрим программу:

```
Program P45;
 { Считывание чисел с клавиатуры }
var i, j : integer;
    x, y : real;
begin
  read(i, j, x, y);
  writeln('Были введены:');
  writeln('i=', i);
  writeln('j=', j);
  writeln('x=', x);
  writeln('y=', y);
end.
```

в которой считываются с клавиатуры значения переменных `i`, `j`, `x`, `y`. После запуска программы на выполнение пользователь набирает:

```
1<ENTER>
2<ENTER>
3.0<ENTER>
4.0<ENTER>
```

На экран будет выведено:

```
Были введены:
i=1
j=2
x=3.0000000000E+00
y=4.0000000000E+00
```

Если данные вводить одной строкой, то результат не изменится:

```
1 2 3.0 4.0<ENTER>
```

В случае необходимости целые числа, введенные пользователем, переводятся в вещественные значения.

Например, в программе `P45` пользователь может набрать на клавиатуре

```
1 2 3 4<ENTER>
```

Процедура `readln` считывает данные точно так же, как и процедура `read`. Однако после считывания последнего значения оставшиеся символы текущей строки игнорируются. В качестве примера рассмотрим программу `P46`:

```
Program P46;
 { Использование процедуры readln }
var i, j : integer;
    x, y : real;
```

```
begin
  writeln('Использование процедуры read');
  read(i, j);
  read(x, y);
  writeln('Были введены:');
  writeln('i=', i, ' j=', j, ' x=', x, ' y=', y);
  writeln('Использование процедуры readln');
  readln(i, j);
  readln(x, y);
  writeln('Были введены:');
  writeln('i=', i, ' j=', j, ' x=', x, ' y=', y);
end.
```

При выполнении операторов

```
read(i, j);
read(x, y);
```

числовые значения из строки

```
1 2 3 4<ENTER>
```

введенной пользователем, будут присвоены соответственно переменным `i`, `j`, `x`, `y`. При выполнении оператора

```
readln(i, j)
```

числовые значения `1` и `2` из строки

```
1 2 3 4<ENTER>
```

будут присвоены переменным `i` и `j`. Числа `3` и `4` игнорируются. Затем компьютер выполняет оператор:

```
readln(x, y)
```

т. е. ожидается набор значений для `x` и `y`. Отметим, что при вызове процедуры `readln` без параметров компьютер ожидает нажатия клавиши `<ENTER>`. Такой вызов процедуры используется с целью приостановки выполнения программы, предоставляя тем самым пользователю возможность прочитать результаты, выведенные до этого на экран.

Для того чтобы сообщить пользователю, какие данные нужно вводить, рекомендуется выводить на экран соответствующие комментарии.

Примеры:

1) `write('Введите два числа:'); readln(x, y);`

2) `write('Введите целое число:'); readln(i);`

3) `write('x='); readln(x);`

4) `write('Ответьте да/D или нет/N:'); readln(c);`

Вопросы и упражнения

❶ Как разделяются числовые данные, вводимые с клавиатуры?
❷ Каковы различия между процедурами `read` и `readln`?
❸ Дана следующая программа:

```
Program P47;
var i : integer;
    c : char;
    x : real;
begin
 readln(i);
 readln(c);
 readln(x);
 writeln('i=', i);
 writeln('c=', c);
 writeln('x=', x);
 readln;
end.
```

Определите результаты, которые будут выведены на экран при вводе следующих данных:

a)
```
1
2
3
```

b)
```
1 2 3
5 6 7
8 9 0
```

c)
```
123
456
789
```

d)
```
123  456  789
abc def ghi
890 abc def
```

3.9. Пустой оператор

Выполнение данного оператора никак не влияет на значения переменных, используемых в программе. Синтаксис пустого оператора:

<*Пустой оператор*> ::=

Таким образом, в тексте программы пустой оператор ничем не представляется. Так как операторы программы разделяются с помощью «`;`», присутствие пустого оператора отмечается появлением этого символа.

Например, в тексте:

```
x:=4;;;; y:=x+1
```

есть 5 операторов, 3 из которых – пустые.

Обычно пустой оператор используется на этапах разработки и отладки сложных программ. Хотя пустой оператор не выполняет никаких действий, его (точнее, символа «`;`») присутствие или отсутствие может повлиять на ход программы.

3.10. Условный оператор `if`

Простой оператор выбора **`if`** выполняет одно из двух возможных действий в зависимости от значения некоторого условия – логического выражения. Синтаксис данного оператора:

<*Оператор* **`if`**> ::= **`if`** <*Логическое выражение*> **`then`** <*Оператор*>
[**`else`** <*Оператор*>]

Синтаксическая диаграмма рассматриваемого оператора представлена на *рис. 3.7*. Логическое выражение, входящее в состав оператора **`if`**, называется **условием**.

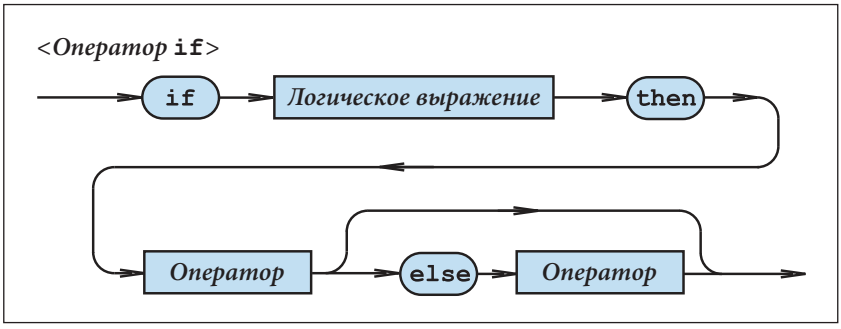


Рис. 3.7. Синтаксическая диаграмма <*Оператор* **`if`**>

Выполнение оператора **`if`** начинается с проверки условия. Если результатом проверки является `true`, то выполняется оператор, стоящий после ключевого слова **`then`**. Если условие принимает значение `false`, то выполняется оператор, стоящий после ключевого слова **`else`** (если оно есть), или управление передается оператору, следующему непосредственно за оператором **`if`**.

В следующей программе оператор **`if`** используется для определения максимального из двух чисел `x` и `y`, считываемых с клавиатуры.

```
Program P48;
{ Определение максимального из двух чисел }
var x, y, max : real;
begin
 writeln('Введите два числа:');
 write('x='); readln(x);
 write('y='); readln(y);
 if x>=y then max:=x else max:=y;
 writeln('max=', max);
 readln;
end.
```

Следующая программа переводит римские цифры: *I* (один), *V* (пять), *X* (десять), *L* (пятьдесят), *C* (сто), *D* (пятьсот), *M* (тысяча), считываемые с клавиатуры, в соответствующие им числа в десятичной системе счисления.

```
Program P49;
 { Перевод римских цифр }
var i : integer; c : char;
begin
 i:=0;
 writeln('Введите одну из римских цифр');
 writeln('I, V, X, L, C, D, M');
 readln(c);
 if c='I' then i:=1;
 if c='V' then i:=5;
 if c='X' then i:=10;
 if c='L' then i:=50;
 if c='C' then i:=100;
 if c='D' then i:=500;
 if c='M' then i:=1000;
 if i=0 then writeln(c, ' - не является римской цифрой')
        else writeln(i);
 readln;
end.
```

Отметим, что в языке ПАСКАЛЬ символ "`;`" не является частью оператора, а используется в качестве разделителя. Следовательно, если в операторе:

```
if B then S
```

перед S поставить пустой оператор

```
if B then; S
```

тогда S не будет входить в состав условного оператора. В таком случае S будет выполняться независимо от значения B.

Если в операторе

```
if B then I else J
```

после I поставить символ "`;`", то получим синтаксически неправильно составленную программу:

```
if B then I; else J
```

В данном случае текст **`else J`** интерпретируется как оператор, стоящий после условного оператора.

Вопросы и упражнения

❶ Для чего необходим оператор **`if`**?

❷ Какие значения будет принимать переменная x после выполнения каждого из следующих операторов? Подразумевается, что `a = 18, b = -15, p = true`.

a) `if a>b then x:=1 else x:=4;`

b) `if a<b then x:=15 else x:=-21;`

c)
```
if p then x:=32 else x:=638;
```

d)
```
if not p then x:=0 else x:=1;
```

e)
```
if (a<b) and p then x:=-1 else x:=1;
```

f)
```
if (a>b) or p then x:=-6 else x:=-5;
```

g)
```
if not (a>b) then x:=19 else x:=-2;
```

h)
```
if (a=b) or p then x:=89 else x:=-15.
```

❸ Напишите программу, которая вычисляет значение одной из следующих функций:

a) $y = \begin{cases} 2x, x \geq 0; \\ \dfrac{x}{2}, x < 0; \end{cases}$

b) $y = \begin{cases} x+3, x > 5; \\ x-3, x \leq 5; \end{cases}$

c) $y = \begin{cases} x, x \geq 3; \\ x+4, x < 3; \end{cases}$

d) $y = \begin{cases} x, |x| > 5; \\ 2x, |x| \leq 5. \end{cases}$

Например, для $y = \begin{cases} x+6, x > 4; \\ x-3, x \leq 4; \end{cases}$ получаем:

```
Program P50;
var x, y : real;
begin
 write('x='); readln(x);
 if x>4 then y:=x+6 else y:=x-3;
 writeln('y=', y);
 readln;
end.
```

❹ Какие результаты выведет на экран следующая программа?

```
Program P51;
var x, y : real;
begin
 write('x='); readln(x);
 y:=x;
 if x>0 then; y:=2*x;
 writeln('y=', y);
 readln;
end.
```

❺ Прокомментируйте сообщения, выводимые на экран в процессе компиляции программы P52:

```
Program P52;
 { Eroare }
var x, y : real;
```

```
begin
 write('x=');
 readln(x);
 if x>4 then y:=x+6;
        else y:=x-3;
 writeln('y=', y); readln;
end.
```

❻ Напишите программу, которая переводит десятичные числа 1, 5, 10, 50, 100, 500 и 1000, считываемые с клавиатуры, в римские.

3.11. Оператор выбора case

Оператор выбора **case** состоит из выражения, называемого **переключателем** (**селектором**), списка констант и соответствующего ему списка операторов. Каждому оператору из списка соответствует одна или несколько констант выбора. Синтаксис рассматриваемого оператора:

<*Оператор* **case**> ::= **case** <*Выражение*> **of** [<*Вариант*>{**;** <*Вариант*>}] [**;**] **end**

<*Вариант*> ::= <*Константа*> {**,** <*Константа*>} **:** <*Оператор*>

Соответствующие синтаксические диаграммы представлены на *рис. 3.8*.

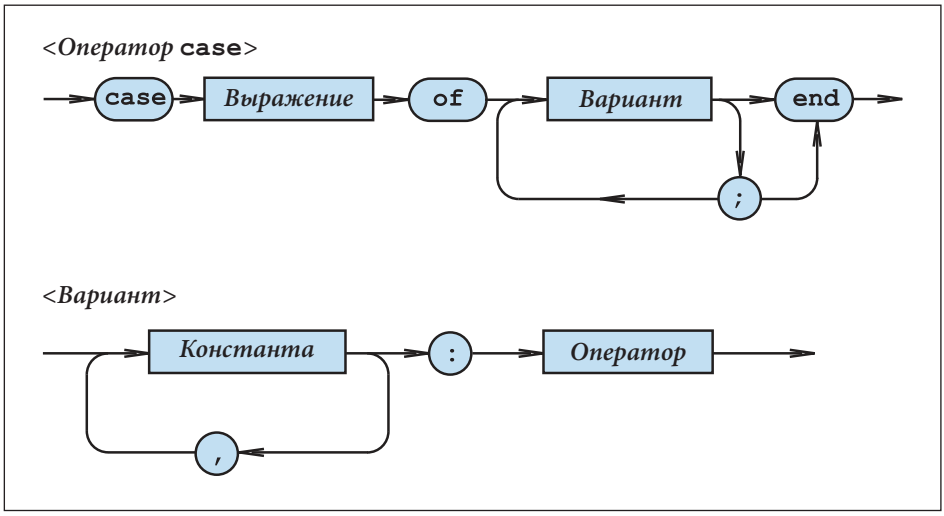


Рис. 3.8. Синтаксические диаграммы оператора **case**

Селектор должен относиться к порядковому типу. Константы выбора должны быть совместимыми с типом селектора и не могут повторяться.

Пример:

```
var i : integer; c : char; a, b, y : real;
```

1)
```
case i of
  0, 2, 4, 6, 8 : writeln('Четная цифра');
  1, 3, 5, 7, 9 : writeln('Нечетная цифра');
end;
```

2)
```
case c of
  '+' : y:=a+b;
  '-' : y:=a-b;
  '*' : y:=a*b;
  '/' : y:=a/b;
end;
```

Выполнение оператора **case** начинается с проверки селектора. Если селектор принимает одно из значений констант выбора, то выполняется оператор, соответствующий этой константе.

В следующей программе оператор **case** используется для перевода римских цифр в десятичные.

```
Program P53;
 { Перевод римских цифр в десятичные }
var i : integer; c : char;
begin
 i:=0;
 writeln('Введите одну из римских цифр');
 writeln('I, V, X, L, C, D, M');
 readln(c);
 case c of
   'I' : i:=1;
   'V' : i:=5;
   'X' : i:=10;
   'L' : i:=50;
   'C' : i:=100;
   'D' : i:=500;
   'M' : i:=1000;
 end;
 if i=0  then writeln(c, ' - не является римской цифрой')
         else writeln(i);
 readln;
end.
```

Отметим, что в некоторых версиях языка синтаксис и семантика оператора **case** были изменены. Список вариантов может включать оператор, которому предшествует ключевое слово **else** (в некоторых версиях **otherwise**). Константы выбора можно заменить интервалами вида

<Константа>..<Константа>

Пример (Turbo PASCAL 7.0):

```
Program P54;
 { Модель карманного калькулятора }
var a, b : real;
    c : char;
begin
 write('a='); readln(a);
 write('b='); readln(b);
 write('Код операции '); readln(c);
 case c of
   '+' : writeln('a+b=', a+b);
   '-' : writeln('a-b=', a-b);
   '*' : writeln('a*b=', a*b);
   '/' : writeln('a/b=', a/b);
 else writeln('Недопустимый код операции');
 end;
 readln;
end.
```

Вопросы и упражнения

❶ Укажите на синтаксических диаграммах *рис. 3.8* пути, которые соответствуют операторам **case** из программ P53 и P54.

❷ Как выполняется оператор **case**? Каким должен быть тип селектора?

❸ Какие константы можно использовать в качестве констант выбора?

❹ Замените оператор **case** программы P54 последовательностью эквивалентных ему операторов **if**.

❺ Используя оператор **case**, напишите программу, которая переводит десятичные числа 1, 5, 10, 50, 100, 500, 1000, считываемые с клавиатуры, в римские.

❻ Что появится на экране в процессе выполнения программы P55?

```
Program P55;
type Semnal=(Rosu, Galben, Verde);
var s : Semnal;
begin
 s:=Verde;
 s:=pred(s);
 case s of
   Rosu : writeln('СТОП');
   Galben : writeln('ВНИМАНИЕ');
   Verde : writeln('СТАРТ');
 end;
 readln;
end.
```

❼ Прокомментируйте следующие программы:

```
Program P56;
 { Eroare }
var x : real;
begin
 writeln('x='); readln(x);
 case x of
   0,2,4,6,8 : writeln('Четная цифра');
   1,3,5,7,9 : writeln('Нечетная цифра');
 end;
 readln;
end.
```

```
Program P57;
 { Eroare }
var i : 1..4;
begin
 write('i='); readln(i);
 case i of
   1 : writeln('Один');
   2 : writeln('Два');
   3 : writeln('Три');
   4 : writeln('Четыре');
   5 : writeln('Пять');
 end;
 readln;
end.
```

```
Program P58;
 { Eroare }
type Semnal = (Rosu, Galben, Verde);
     Culoare = (Albastru, Portocaliu);
var s : Semnal;
    c : Culoare;
begin
 { ... }
 case s of
   Rosu : writeln('СТОП');
   Galben : writeln('ВНИМАНИЕ');
   Verde : writeln('СТАРТ);
   Albastru : writeln('ПАУЗА');
 end;
 { ... }
end.
```

3.12. Оператор for

Оператор **for** предназначен для повторного выполнения другого оператора в зависимости от значения управляющей переменной. Синтаксис рассматриваемого оператора:

<*Оператор* **for**> ::= **for** <*Переменная*> **:=** <*Выражение*> <*Шаг*> <*Выражение*> **do** <*Оператор*>

<*Шаг*> ::= **to** | **downto**

Соответствующие синтаксические диаграммы представлены на *рис. 3.9.*

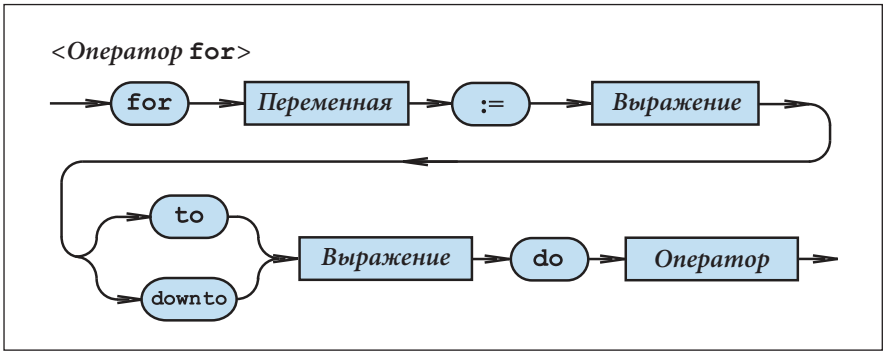


Рис. 3.9. Синтаксическая диаграмма оператора **for**

Переменная, находящаяся после ключевого слова **for**, называется управляющей переменной или параметром цикла. Эта переменная должна принадлежать некоторому порядковому типу.

Значения выражений, входящих в состав оператора **for**, должны быть совместимыми, с точки зрения присваивания, с параметром цикла. Эти выражения проверяются один раз в начале цикла. Первое выражение указывает исходное значение параметра цикла, а второе – конечное значение.

Оператор, стоящий после ключевого слова **do**, выполняется для каждого значения из диапазона, определяемого начальным и конечным значениями.

Если в операторе **for** используется шаг **to**, то значение параметра цикла увеличивается при каждом повторении, переходя к значению, следующему за текущим. Если исходное значение больше конечного, оператор, расположенный после ключевого слова **do**, не выполняется ни разу. Если в операторе **for** используется шаг **downto**, то значение параметра цикла уменьшается при каждом повторении, переходя к значению, предшествующему текущему. Если начальное значение меньше конечного, то оператор, расположенный после ключевого слова **do**, не выполняется ни разу.

Пример:

```
Program P59;
 { Оператор for }
var i : integer;
    c : char;
```

```
begin
  for i:=0 to 9 do write(i:2);
  writeln;
  for i:=9 downto 0 do write(i:2);
  writeln;
  for c:='A' to 'Z' do write(c:2);
  writeln;
  for c:='Z' downto 'A' do write(c:2);
  writeln;
  readln;
end.
```

Результаты, выводимые на экран:

```
0 1 2 3 4 5 6 7 8 9
9 8 7 6 5 4 3 2 1 0
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
Z Y X W V U T S R Q P O N M L K J I H G F E D C B A
```

Значения параметра цикла не могут быть изменены внутри цикла, т. е.

1) параметру цикла не присваиваются никакие значения;

2) параметр цикла не может быть параметром цикла другого вложенного оператора **for**;

3) нельзя вызывать процедуры `read` и `readln`, в которых указывается параметр цикла.

После выхода из оператора **for** значение параметра цикла не определено, за исключением случая, когда выход из цикла осуществляется принудительно, через оператор безусловного перехода **goto**.

Оператор **for** используется для программирования итеративных алгоритмов, в которых число повторений известно. В качестве примера рассмотрим программы `P60`, `P61` и `P62`, которые вычисляют соответственно $n!$, x^n и сумму $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+...+\frac{1}{n}$.

```
Program P60;
 { Вычисление факториала }
var n, i, f : 0..MaxInt;
begin
  write('n='); readln(n);
  f:=1;
  for i:=1 to n do f:=f*i;
  writeln('n!=', f);
  readln;
end.
```

```
Program P61;
 { Вычисление x в степени n }
```

```
var x, y : real;
    n, i : 0..MaxInt;
begin
  write('x='); readln(x);
  write('n='); readln(n);
  y:=1;
  for i:=1 to n do y:=y*x;
  writeln('y=', y);
  readln;
end.
```

```
Program P62;
 { Вычисление суммы 1 + 1/2 + 1/3 + ... + 1/n }
var n, i : 1..MaxInt;
    s : real;
begin
  write('n=');
  readln(n);
  s:=0;
  for i:=1 to n do s:=s+1/i;
  writeln('s=', s);
  readln;
end.
```

Вопросы и упражнения

❶ Укажите на синтаксической диаграмме *рис. 3.9* пути, которые соответствуют операторам **for** из программы P59.

❷ Как выполняется оператор **for**?

❸ Что выведет на экран программа P63?

```
Program P63;
type Zi = (L, Ma, Mi, J, V, S, D);
var z : Zi;
begin
  for z:=L to S do writeln(ord(z));
  readln;
  for z:=D downto Ma do writeln(ord(z));
  readln;
end.
```

❹ Даны описания:

```
var i, j, n : integer;
    x, y : real;
    c : char;
```

Какие из следующих операторов являются синтаксически правильными?

a) `for i:=-5 to 5 do j:=i+3;`

b) `for i:=-5 to 5 do i:=j+3;`

c) `for j:=-5 to 5 do i:=j+3;`

d) `for i:=1 to n do y:=y/i;`

e) `for x:=1 to n do y:=y/x;`

f) `for c:='A' to 'Z' do writeln(ord(c));`

g) `for c:='Z' downto 'A' do writeln(ord(c));`

h) `for i:=-5 downto -10 do readln(i);`

i) `for i:=ord('A') to ord('A')+ 9 do writeln(i);`

j) `for c:='0' to '9' do writeln(c, ord(c):3);`

k) `for j:=i/2 to i/2+10 do writeln(j).`

❺ Даны описания

```
var i, m, n : integer;
```

Сколько раз будут выполнены процедуры `writeln(i)` и `writeln(2*i)`, входящие в состав операторов:

```
for i:=m to n do writeln(i);
for i:=m to n do writeln(2*i);
```

если:

a) `m=1, n=5;`

b) `m=3, n=5;`

c) `m=3, n=3;`

d) `m=5, n=3?`

❻ Напишите программу, которая выводит на экран коды символов `'A'`, `'B'`, `'C'`,… , `'Z'`.

❼ Вычислите для первых *n* элементов:

a) $1 + 3 + 5 + 7 + \ldots$ и $1 \cdot 3 \cdot 5 \cdot 7 \cdot \ldots$;

b) $2 + 4 + 6 + 8 + \ldots$ и $2 \cdot 4 \cdot 6 \cdot 8 \cdot \ldots$;

c) $3 + 6 + 9 + 12 + \ldots$ и $3 \cdot 6 \cdot 9 \cdot 12 \cdot \ldots$;

d) $4 + 8 + 12 + 16 + \ldots$ и $4 \cdot 8 \cdot 12 \cdot 16 \cdot \ldots$.

Например: При $n=3$ имеем $1 + 3 + 5 = 9$; $1 \cdot 3 \cdot 5 = 15$.

❽ Вычислите сумму первых *n* элементов:

$$1-\frac{1}{2}+\frac{1}{3}-\frac{1}{4}+\frac{1}{5}-\frac{1}{6}+\ldots.$$

Указание. Используйте внутри цикла оператор **`if`** `odd(…)` **`then`** … **`else`** … .

3.13. Составной оператор

Синтаксис составного оператора:

<*Составной оператор*> ::= **begin** <*Оператор*> {**;** <*Оператор*>} **end**

Соответствующая синтаксическая диаграмма представлена на *рис. 3.10*.

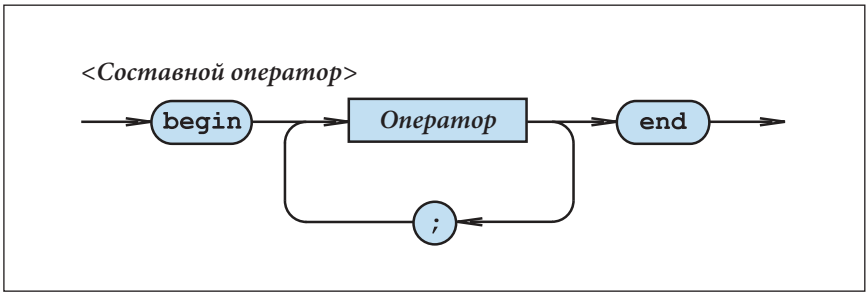


Рис. 3.10. Синтаксическая диаграмма <*Составной оператор*>

Примеры:

1)
```
begin
  a:=x+12;
  p:=q and r;
  writeln(p)
end;
```

2)
```
begin
  write('x=');
  readln(x)
end;
```

Ключевые слова **begin** и **end** выступают в роли "скобок". Последовательность операторов, заключенных в данные скобки, является, с точки зрения языка, одним оператором. Таким образом, составной оператор используется для того, чтобы несколько операторов поместить в те места программы, где разрешается наличие только одного оператора (см. операторы **if**, **case**, **for** и др.).

Примеры:

1)
```
if a>0 then begin x:=a+b; y:=a*b end
       else begin x:=a-b; y:=a/b end;
```

2)
```
case c of
  '+' : begin y:=a+b; writeln('Adunarea') end;
  '-' : begin y:=a-b; writeln('Scaderea') end;
  '*' : begin y:=a*b; writeln('Inmultirea') end;
  '/' : begin y:=a/b; writeln('Impartirea') end;
end;
```

3)
```
for i:=1 to n do
  begin
    write('x=');
    readln(x);
    s:=s+x
  end;
```

Отметим, что тело любой программы является составным оператором, так как оно представляет собой последовательность операторов, заключенных в "скобки" **begin** и **end**.

Так как символ "**;**" не заканчивает, а разделяет операторы, то присутствие его перед ключевым словом **end** не обязательно. Однако многие программисты вставляют данный символ с целью продолжить список операторов в случае необходимости. Напоминаем, что дополнительное появление символа "**;**" означает вставку пустого оператора.

Для того чтобы программы были удобными для чтения, слова **begin** и **end** пишутся строго одно под другим, а операторы внутри "скобок" смещаются на несколько позиций вправо. Если составной оператор **begin...end** включается в состав других операторов (**if**, **case**, **for** и др.), то ключевые слова **begin** и **end** смещаются вправо.

В качестве примера рассмотрим программу P64, в которой вычисляется среднее арифметическое *n* чисел, считываемых с клавиатуры.

```
Program P64;
 { Среднее арифметическое n чисел }
var x, suma, media : real;
    i, n : integer;
begin
 write('n='); readln(n);
 suma:=0;
 writeln('Введите ', n, ' чисел:');
 for i:=1 to n do
   begin
    write('x='); readln(x);
    suma:=suma+x;
   end;
 if n>0 then
          begin
            media:=suma/n;
            writeln('media=', media);
          end
        else writeln('media=*****');
 readln;
end.
```

Вопросы и упражнения

❶ Для чего необходим составной оператор?

❷ Укажите на синтаксической диаграмме *рис. 3.10* пути, которые соответствуют составному оператору из программы P64.

❸ Напишите программу, которая считывает с клавиатуры *n* чисел и затем выводит на экран:
а) сумму и среднее арифметическое считанных чисел;
б) сумму и среднее арифметическое положительных чисел;
в) сумму и среднее арифметическое отрицательных чисел.

❹ Напишите программу, которая считывает с клавиатуры *n* символов и затем выводит на экран:
a) количество считанных десятичных цифр;
b) количество четных цифр;
c) количество нечетных цифр;
d) количество считанных букв;
e) количество гласных;
f) количество согласных.
Вводимые символы разделяются нажатием клавиши <ENTER>. Предполагается, что будут вводиться десятичные цифры 0, 1, 2, ... 9 и прописные буквы латинского алфавита *A*, *B*, *C*, ... *Z*.

3.14. Оператор while

Оператор **while** состоит из логического выражения, которое управляет многократным выполнением других операторов. Синтаксис рассматриваемого оператора:

<*Оператор* **while**> ::= **while** <*Логическое выражение*> **do** <*Оператор*>

Синтаксическая диаграмма представлена на *рис. 3.11*.

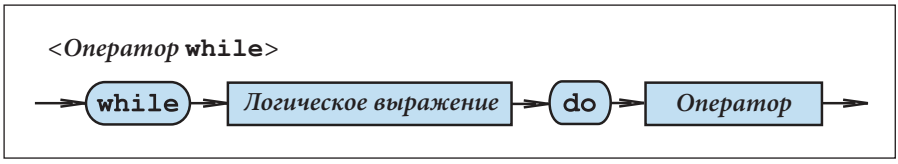


Рис. 3.11. Синтаксическая диаграмма оператора **while**

Примеры:

1)
```
while x>0 do x:=x-1;
```

2)
```
while x<3.14 do
  begin
    x:=x+0.001;
    writeln(sin(x));
  end;
```

3)
```
while p do
  begin
    x:=x+0.001;
    y:=10*x;
    p:=y<1000;
  end;
```

Оператор, расположенный после ключевого слова **do**, выполняется многократно до тех пор, пока логическое выражение принимает значение `true`. Как только логическое значение принимает значение `false`, оператор, стоящий после ключевого слова **do**, перестает выполняться. Логическое выражение должно иметь наиболее простой вид, так как оно проверяется при каждой итерации.

Обычно оператор **while** используется для организации повторных вычислений в циклах, где управляющей является переменная типа `real`.

В следующей программе оператор **while** используется для вывода значений функции $y=2x$. Аргумент x принимает значения от x_1 до x_2 с шагом Δx.

```
Program P65;
 { Таблица функции y=2*x }
var x, y, x1, x2, deltaX : real;
begin
  write('x1='); readln(x1);
  write('x2='); readln(x2);
  write('deltaX='); readln(deltaX);
  writeln('x':10, 'y':20);
  writeln;
  x:=x1;
  while x<=x2 do
    begin
      y:=2*x;
      writeln(x:20, y:20);
      x:=x+deltaX;
    end;
 readln;
end.
```

Оператор **while** обычно используется в тех случаях, когда трудно определить число повторений некоторой последовательности операторов.

В качестве примера рассмотрим программу P66, которая выводит на экран среднее арифметическое положительных чисел, считываемых с клавиатуры.

```
Program P66;
 { Среднее арифметическое положительных чисел,
   считываемых с клавиатуры }
var x, suma : real;
    n : integer;
begin
  n:=0;
```

```
  suma:=0;
  writeln('Введите положительные числа:');
  readln(x);
  while x>0 do
    begin
      n:=n+1;
      suma:=suma+x;
      readln(x);
    end;
  writeln('Было введено ', n, ' положительных чисел.');
  if n>0 then writeln('среднее=', suma/n)
         else writeln('среднее=*****');
  readln;
end.
```

Отметим, что число повторений составного оператора **begin...end**, входящего в состав оператора **while**, заранее неизвестно. Оператор **while** завершает свое выполнение, когда пользователь вводит число $x \le 0$.

Вопросы и упражнения

❶ Как выполняется оператор **while**?

❷ Укажите на синтаксических диаграммах *рис. 3.11* пути, которые соответствуют операторам **while** в программах P65 и P66.

❸ Используя оператор **while**, напишите программу, которая выводит на экран значения функции $y = f(x)$ для аргумента, принимающего значения от x_1 до x_2 с шагом Δx:

a) $y = \frac{x}{3} + 2;$

b) $y = \frac{x}{2};$

c) $y = 3x - 4;$

d) $y = 4x - 13.$

❹ Пользователь вводит с клавиатуры целые положительные числа, разделяемые нажатием клавиши <ENTER>. Признаком конца последовательности является число 0. Напишите программу, которая выводит на экран:
a) сумму и среднее арифметическое четных чисел;
b) сумму и среднее арифметическое нечетных чисел.

❺ Напишите программу, которая выводит на экран значения функции $y = f(x)$. Аргумент x принимает значения от x_1 до x_2 с шагом Δx:

a) $y = \begin{cases} x, & x > 3; \\ 2x, & x \le 3; \end{cases}$

b) $y = \begin{cases} 6x, & x \ge 0; \\ 4x, & x < 0; \end{cases}$

c) $y = \begin{cases} x + 6, & x > 5; \\ x - 6, & x \le 5; \end{cases}$

d) $y = \begin{cases} 3 - x, & x > 4; \\ 3 + x, & x \le 4. \end{cases}$

Пример: $y = \begin{cases} x+1, & x > 8; \\ x-2, & x \le 8. \end{cases}$

```
Program P67;
 { Таблица функции }
var x, y, x1, x2, deltaX : real;
begin
  write('x1='); readln(x1);
  write('x2='); readln(x2);
  write('deltaX='); readln(deltaX);
  writeln('x':10, 'y':20);
  writeln;
  x:=x1;
  while x<=x2 do
    begin
      if x>8 then y:=x+1 else y:=x-2;
      writeln(x:20, y:20);
      x:=x+deltaX;
    end;
  readln;
end.
```

❻ Оператор цикла

```
for i:=i1 to i2 do writeln(ord(i))
```

эквивалентен следующей последовательности операторов:

```
i:=i1;
while i<=i2 do
  begin
    writeln(ord(i));
    i:=succ(i);
  end.
```

Напишите эквивалентную последовательность операторов для оператора цикла:

```
for i:=i1 downto i2 do writeln(ord(i))
```

❼ Даны описания:

```
var x1, x2, deltaX : real;
    i, n : integer;
```

Какие из следующих последовательностей операторов эквивалентны?

a)
```
x:=x1;
while x<=x2 do
begin
  writeln(x);
  x:=x+deltaX;
end;
```

b)
```
n:=trunc((x2-x1)/deltaX)+1;
x:=x1;
for i:=1 to n do
```

```
  begin
    writeln(x);
    x:=x+deltaX;
  end;
```

c)
```
n:=round((x2-x1)/deltaX)+1;
x:=x1;
  for i:=1 to n do
    begin
      writeln(x);
      x:=x+deltaX;
    end;
```

Аргументируйте ваш ответ.

3.15. Оператор `repeat`

Оператор **`repeat`** указывает на многократное выполнение последовательности операторов в зависимости от значения некоторого логического выражения.

Синтаксис рассматриваемого оператора:

<Оператор **`repeat`***>* ::= **`repeat`** *<Оператор>* {**`;`** *<Оператор>*} **`until`** *<Логическое выражение>*

Синтаксическая диаграмма представлена на *рис. 3.12*.

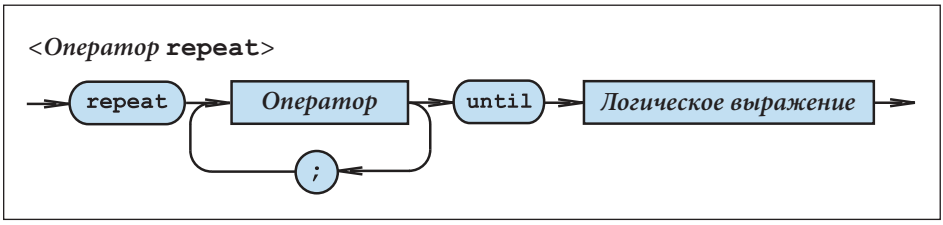


Рис. 3.12. Синтаксическая диаграмма оператора **`repeat`**

Примеры:

1)
```
repeat x:=x-1 until x<0;
```

2)
```
repeat
  y:=y+delta;
  writeln(y)
until y>20.5;
```

3)
```
repeat
  readln(i);
  writeln(odd(i))
until i=0;
```

Операторы, расположенные между ключевыми словами **repeat** и **until**, выполняются многократно до тех пор, пока логическое выражение сохраняет значение false. Как только логическое выражение становится истинным, управление переходит к следующему оператору. Очевидно, что операторы, стоящие между ключевыми словами **repeat** и **until**, будут выполнены по крайней мере один раз, так как логическое выражение проверяется лишь после выполнения указанной последовательности операторов.

Как правило, оператор **repeat** используется вместо оператора **while** в тех случаях, когда проверка логического выражения, управляющего повторением, должна производиться после выполнения соответствующей последовательности операторов.

Программа, приведенная ниже, выводит на экран сообщение о четности чисел, считываемых с клавиатуры.

```
Program P68;
 { Четность чисел, считываемых с клавиатуры }
var i : integer;
begin
 writeln('Введите целые числа:');
 repeat
   readln(i);
   if odd(i) then writeln(i:6, ' - нечетное число')
             else writeln(i:6, ' - четное число');
 until i=0;
 readln;
end.
```

Выполнение оператора **repeat** завершается, когда пользователь вводит $i = 0$.

Очень часто оператор **repeat** применяется для проверки правильности данных, вводимых с клавиатуры. Например, предположим, что необходимо написать программу, считывающую с клавиатуры вещественное число x и выдающую на экран квадратный корень $y = \sqrt{x}$. Очевидно, что отрицательные значения переменной x являются недопустимыми.

```
Program P69;
 { Вычисление квадратного корня }
var x, y : real;
begin
  repeat
    write('Введите неотрицательное число x=');
    readln(x);
  until x>=0;
  y:=sqrt(x);
  writeln('Квадратный корень y=', y);
  readln;
end.
```

При выполнении программы P69, компьютер предлагает пользователю ввести неотрицательное число. В случае, если пользователь, по ошибке, введет отрицательное число, операторы **write** и **readln** из состава оператора **repeat** будут выполнены

заново. Циклический процесс продолжится до тех пор, пока пользователь не введет правильное число.

Из данных примеров видно, что оператор **repeat** используется тогда, когда число повторений некоторой последовательности операторов сложно предугадать.

Вопросы и упражнения

❶ Как выполняется оператор **repeat**?

❷ Укажите на синтаксических диаграммах *рис. 3.12* пути, которые соответствуют операторам **repeat** из программ P68 и P69.

❸ Даны следующие операторы:

a)
```
repeat
   <Оператор 1>;
   <Оператор 2>;
   ...
   <Оператор n>;
until p
```

b)
```
while not p do
begin
   <Оператор 1>;
   <Оператор 2>;
   ...
   <Оператор n>;
end.
```

Эквивалентны ли эти операторы? Аргументируйте ваш ответ.

❹ Напишите программу, которая считывает с клавиатуры последовательность символов и выводит на экран:
a) количество считанных десятичных цифр;
b) количество четных цифр;
c) количество нечетных цифр.
Вводимые символы разделяются с помощью клавиши <ENTER>. Предполагается, что можно вводить десятичные цифры 0, 1, 2, ... 9 и символ *, который указывает конец последовательности.

❺ Напишите программу, которая считывает с клавиатуры вещественное число x и выдает на экран значение выражения $\frac{1}{x}$. Очевидно, значение $x = 0$ является недопустимым. Для проверки вводимых с клавиатуры данных воспользуйтесь оператором **repeat**.

❻ Эквивалентен ли оператор:
```
for i:=i1 to i2 do writeln(ord(i))
```
последовательности операторов:
```
i:=i1;
repeat
  writeln(ord(i));
  i:=succ(i);
until i>i2;
```
Аргументируйте ваш ответ.

❼ Напишите программу, которая выводит на экран значения функции $y=f(x)$. Аргумент x принимает значения от x_1 до x_2 с шагом Δx, а цикл организуется с помощью оператора **repeat**.

a) $y = 2x;$ *c)* $y = x - 4;$

b) $y = \frac{x}{3} + 9;$ *d)* $y = \frac{x}{8} - 6.$

❽ Напишите программу, которая считывает с клавиатуры последовательность символов и выводит на экран:

a) количество считанных букв;

b) количество прописных букв;

c) количество строчных букв.

Вводимые символы разделяются с помощью клавиши <ENTER>. Предполагается, что можно вводить строчные и прописные буквы латинского алфавита и символ *, который указывает конец последовательности.

3.16. Оператор goto

Как правило, операторы программы выполняются последовательно в том порядке, в котором они появляются в тексте программы. Оператор безусловного перехода **goto** позволяет изменить этот порядок и продолжить работу в другой части текста программы. Синтаксис рассматриваемого оператора:

<*Оператор* **goto**> ::= **goto** <*Метка*>

Напомним, что метка – это целое число без знака, которое указывает на некоторый оператор программы (*рис. 3.1*). Метки перечисляются в разделе описаний программы после ключевого слова **label**. Синтаксис данного описания:

<*Раздел описания меток*> ::= **label** <*Метка*> {, <*Метка*>};

Синтаксические диаграммы соответствующих грамматических единиц представлены на *рис. 3.13*.

Отметим, что перечисление меток в разделе описаний **label** обязательно.

При выполнении оператора **goto** управление передается оператору, помеченному соответствующей меткой.

В качестве примера представим программу P70, которая вычисляет значение функции

$$y = \begin{cases} x, & x \geq 0; \\ 2x, & x < 0. \end{cases}$$

Значения аргумента x считываются с клавиатуры.

```
Program P70;
 { Выполнение оператора goto }
label 1, 2;
var x, y : real;
```

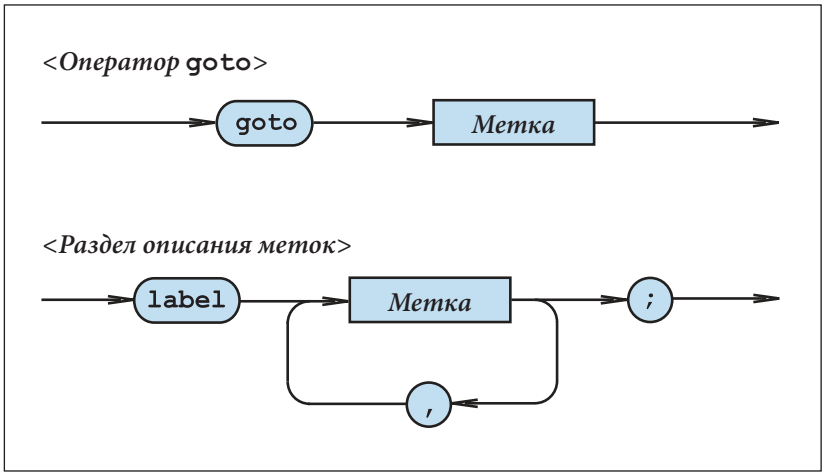


Рис. 3.13. Синтаксические диаграммы <*Оператор* **goto**> и <*Раздел описания меток*>

```
begin
  write('x='); readln(x);
  if x>=0 then goto 1;
  y:=2*x;
  writeln('x<0, y=', y);
  goto 2;
1: y:=x;
  writeln('x>=0, y=', y);
2: readln;
end.
```

Если пользователь набирает любое значение x, $x \geq 0$, то выполняется оператор **goto** 1 и управление передается оператору присваивания:

```
1: y:=x;
```

После этого выполняются операторы:

```
   writeln('x>=0 , y=', y);
2: readln;
```

Если пользователь набирает любое значение x, $x < 0$, то выполняются операторы:

```
y:= 2*x;
writeln('x<0, y=', y);
goto 2;
```

Последний оператор передает управление оператору:

```
2: readln;
```

В любой программе для меток и операторов должны выполняться следующие правила:

1) каждая метка должна быть описана с помощью ключевого слова **label**;

2) каждая метка должна стоять перед одним и только одним оператором;

3) переходы внутри сложных операторов (**if**, **for**, **while**, ... и др.) запрещаются.

Пример:

1)
```
if i>5 then 1: writeln('i>5') else writeln('i<=5');
...
goto 1; {Eroare}
```

2)
```
for i:=1 to 10 do
begin
10: writeln('i=', i);
end;
...
goto 10; {Eroare}
```

При отсутствии операторов **goto** все операторы программы выполняются в том порядке, в котором они появляются в тексте программы. Следовательно, операторы **goto** нарушают соответствие между текстом программы и порядком выполнения операторов, что усложняет разработку, проверку и отладку программ. Таким образом, использование оператора **goto** нежелательно.

Например, программу P70 можно переписать в следующем виде:

```
Program P71;
 { Исключение оператора goto из программы P70 }
var x, y : real;
begin
  write('x=');
  readln(x);
  if x>=0 then
  begin
    y:=x;
    writeln('x>=0, y=', y);
  end
    else
  begin
    y:=2*x;
    writeln('x<0, y=', y);
  end;
 readln;
end.
```

Как правило, оператор **goto** используется в исключительных случаях, например, для уменьшения размеров программы.

Вопросы и упражнения

❶ Для чего необходим оператор **goto**?

❷ Укажите на синтаксических диаграммах *рис. 3.13* пути, которые соответствуют меткам и операторам **goto** из программы P70.

❸ Перепишите следующие программы без использования оператора **goto**:

```
Program P72;
 { Вывод приветствия на экран }
label 1, 2, 3;
var i : 6..23;
begin
  write('Который час?'); readln(i);
  if i>12 then goto 1;
  writeln('Доброе утро!');
  goto 3;
1: if i>17 then goto 2;
  writeln('Добрый день!');
  goto 3;
2: writeln('Добрый вечер!');
3: readln;
end.
```

❹ Прокомментируйте следующую программу:

```
Program P73;
 { Eroare }
label 1;
var i : 1..5;
begin
  i:=1;
1: writeln(i);
  i:=i+1;
  goto 1;
end.
```

❺ Что выведет на экран следующая программа?

```
Program P74;
 { Eroare }
label 1;
var x : real;
begin
  x:=0;
1: writeln(x);
  x:=x+1e-30;
  goto 1;
end.
```

Напомним, что работу программы можно прервать нажатием клавиш <CTRL+C> или <CTRL+BREAK>.

❻ Прокомментируйте следующую программу:

```
Program P75;
 { Eroare }
label 1;
var i : integer;
begin
  i:=1;
while i<=20 do
    begin
      writeln(i);
      1: i:=i+1;
    end;
  goto 1;
end.
```

3.17. Структура программы на языке ПАСКАЛЬ

Программа на языке ПАСКАЛЬ имеет следующую структуру:

<Программа> ::= *<Заголовок программы>*
<Блок>.

Заголовок программы содержит имя программы и, если необходимо, список формальных параметров:

<Заголовок программы> ::= **Program** *<Идентификатор>* [(*<Идентификатор>* {, *<Идентификатор>*})];

Примеры:

1) **Program** A1;

2) **Program** B6(Intrare);

3) **Program** C15(Intrare, Iesire);

Обычно формальные параметры используются для связи программы со средой ввода/вывода.

Блок программы состоит из раздела описаний и раздела операторов:

<Блок> ::= *<Раздел описаний>*
<Раздел операторов>

Синтаксис раздела описаний:

<Раздел описаний> ::= [*<Раздел описания меток>*]
[*<Раздел описания констант>*]
[*<Раздел описания типов>*]

[<*Раздел описания переменных*>]
[<*Раздел описания подпрограмм*>]

Раздел операторов представляет собой составной оператор **`begin...end`**.

Синтаксические диаграммы соответствующих грамматических единиц представлены на *рис. 3.14*.

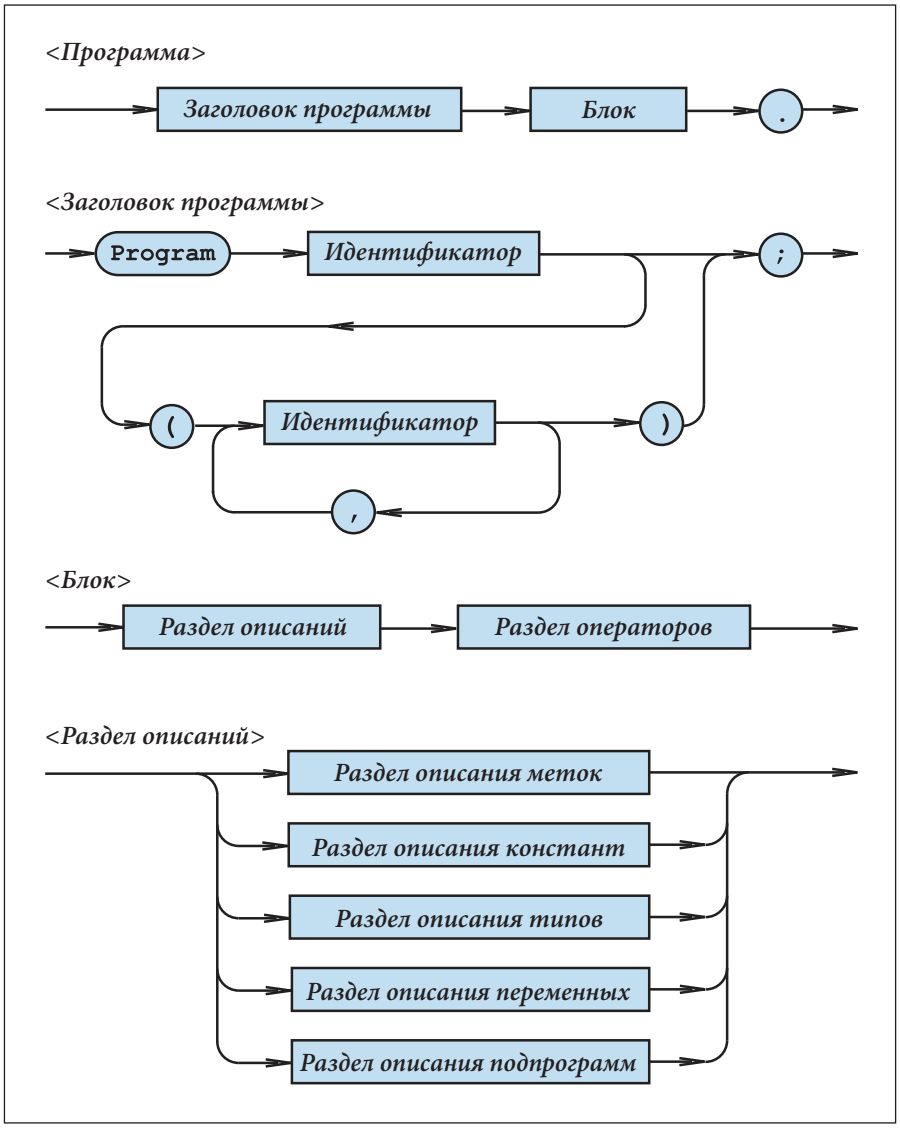


Рис. 3.14. Структура программы на языке ПАСКАЛЬ

Отметим, что конец программы обозначается символом "`.`" (точка).

В заключение в качестве примера представляем программу `P76`, которая вычисляет длину дуги окружности, опирающуюся на угол в α градусов, и площадь соответствующего сектора.

```
Program P76;
  { Длина дуги окружности и площадь
    соответствующего сектора }
label 1, 2;
const Pi=3.141592654;
type  grade=0..360;
var   alfa : grade;
      raza, lungimea, aria : real;
begin
  write('радиус='); readln(raza);
  if raza<0 then goto 1;
  write('alfa='); readln(alfa);
  lungimea:=Pi*raza*alfa/180;
  writeln('длина=', lungimea);
  aria:=Pi*sqr(raza)*alfa/360;
  writeln('площадь=', aria);
  goto 2;
1: writeln('Ошибка: радиус<0');
2: readln;
end.
```

Вопросы и упражнения

❶ Для чего необходим заголовок программы? Как обозначается конец программы?
❷ Укажите на синтаксических диаграммах *рис. 3.14* пути, которые соответствуют грамматическим единицам из программы P76.
❸ Перепишите программу P76 без использования оператора **goto**. Укажите раздел описаний и раздел операторов созданной программы.
❹ Для чего нужны параметры, указываемые в заголовке программы?

Тест для самопроверки № 3

1. Запишите в соответствии с правилами языка ПАСКАЛЬ следующие выражения:

a) $(a + b) - 2ab$;
b) $6a^2 + 15ab - 13b^2$;
c) $(a + b)\,(a - b)$;
d) $2\alpha\beta - 5\pi r$;
e) $\pi r^2 + \alpha\beta^2$;
f) $xy \vee xz$.

2. Запишите в обычном виде выражения, представленные в соответствии с правилами языка ПАСКАЛЬ:

a) `sqr(a)+2/sqr(b)`
b) `2*a/(b+c)`
c) `15*sqrt(a/(a-b))`
d) `not(x and y) or z`
e) `sqr((a+b)/2)`
f) `(x<>0) and (q<p)`

3. Какие из нижеследующих выражений на языке ПАСКАЛЬ ошибочны?

a) `2*a+2*b`

b) `4*sinx+4*cosy`

c) `3*sqr(x)+3/sin(y)`

d) `a+2*-b`

e) `not (q and p)`

f) `2*(+x)+((-y))`

4. Пусть x=1, y=2 и z=3. Вычислите значения нижеследующих выражений:

a) `x+2*y+3*z`

b) `(1+x+y-2)*z`

c) `x*y+y*(-z)`

d) `not(x+y+z>0)`

e) `x*y<y+z`

f) `(x>y)or(2*x<y+z)`

5. Пусть заданы объявления переменных:

```
var  x  : real;
     i  : integer;
     p  : boolean;
     s  : char;
     Zi : (Luni, Marti, Miercuri, Joi, Vineri, Simbata,
           Duminica);
```

Определите тип каждого из следующих выражений:

a) `i mod 9`

b) `i/9`

c) `i+x`

d) `ord(pred(Zi))`

e) `ord(Zi)+trunc(x)`

f) `sqr(ord(s))`

g) `p or (x>i)`

h) `chr(i+ord(p))`

6. Напишите программу на языке ПАСКАЛЬ, которая выводит на экран значение выражения $15i\,(x + y)$. Значения целой переменной *i* и вещественных переменных *x*, *y* считываются с клавиатуры.

7. Даны следующие объявления:

```
type  FunctiaOcupata = (Muncitor, SefDeEchipa, Maistru,
                        SefDeSantier, Director);
      StareaCivila = (Casatorit, Necasatorit);
var  i : integer;
     x : real;
     f : FunctiaOcupata;
     s : StareaCivila;
```

Какие из приведенных ниже операторов являются правильными?

a) `i:=ord(f)+15`

b) `f:=Casatorit`

c) `x:=ord(f)+1`

d) `i:=2*x-15`

e) `s:=pred(s)`

f) `f:=succ(SefdeEchipa)`

8. Напишите программу, которая вычисляет значение функции:

$$y = \begin{cases} 9x + 3x^2, \; x > 15; \\ 3x - 5\sqrt{x + 28}, \; x \le 15. \end{cases}$$

Значение вещественной переменной *x* считывается с клавиатуры.

9. Используемые в Республике Молдова монеты имеют значения 1, 5, 10, 25 или 50 бань. Напишите программу на языке ПАСКАЛЬ, которая считывает с клавиатуры числовое значение монеты и выводит на экран это же значение, выраженное словами. Например, если пользователь вводит „25", то на экран выводится „двадцать пять бань". Если же пользователь вводит отличное от 1, 5, 10, 25 или 50 значение, то на экран выводится сообщение „недопустимое значение".

10. Напишите программу, которая, используя оператор **for**, вычисляет для первых n элементов сумму

$$s = \frac{1}{1} + \frac{1}{2} + \frac{1}{3} + ... + \frac{1}{n}$$

и произведение

$$p = \frac{1}{1} \cdot \frac{1}{2} \cdot \frac{1}{3} \cdot ... \cdot \frac{1}{n}.$$

11. Напишите программу, которая, используя оператор **while**, вычисляет и выводит на экран значения функции

$$y = \begin{cases} 2\sqrt{x + 6}, \; x \ge 4; \\ 3 - abs(x), \; x < 4 \end{cases}$$

для значений аргумента x от x_1 до x_2 с шагом Δx.

12. Специальные сообщения мобильной телефонной связи определяются с помощью следующих металингвистических формул:

<*Цифра*> ::= 0|1|2|3|4|5|6|7|8|9
<*Специальное сообщение*> ::= *{<*Цифра*>}#

Напишите программу которая, используя оператор **repeat**, вычисляет и выводит на экран количество цифр, содержащихся в специальном сообщении. При вводе сообщения с клавиатуры, после каждого символа необходимо нажимать клавишу <ENTER>. Например, если пользователь вводит:

```
*<ENTER>
1<ENTER>
0<ENTER>
4<ENTER>
#<ENTER>
```

на экран должно выводиться число 3.

Глава 4

ОДНОМЕРНЫЕ СОСТАВНЫЕ ТИПЫ ДАННЫХ

4.1. Тип данных *одномерный массив* (array)

Множество значений типа данных **array** состоит из массивов (таблиц). Массивы составляются из фиксированного числа компонент одного и того же типа, который называется **базовым**. Ссылка на компоненты осуществляется с помощью **индексов**.

Тип данных *одномерный массив* определяется конструкцией вида

```
type <Имя типа> = array [T1] of T2;
```

где T_1 – тип индекса, который должен быть порядковым, а T_2 – тип компонент (базовый), который может быть любым.

Примеры:

1)
```
type  Vector = array [1..5] of real;
var   x : Vector;
```

2)
```
type  Zi = (L, Ma, Mi, J, Vi, S, D);
      Venit = array [Zi] of real;
var   v : Venit;
      z : Zi;
```

3)
```
type  Ora = 0..23;
      Grade = -40..40;
      Temperatura = array [Ora] of Grade;
var   t : Temperatura;
      h : Ora;
```

Структура данных, используемых в этих примерах, представлена на *рис. 4.1*.

Доступ к компонентам переменной типа *одномерный массив* осуществляется явно через имя переменной, за которой следует соответствующий индекс, заключенный в квадратные скобки.

Примеры:

1) `x[1], x[4];`

2) `v[L], v[Ma], v[J];`

3) `t[0], t[15], t[23];`

4) `v[z], t[h].`

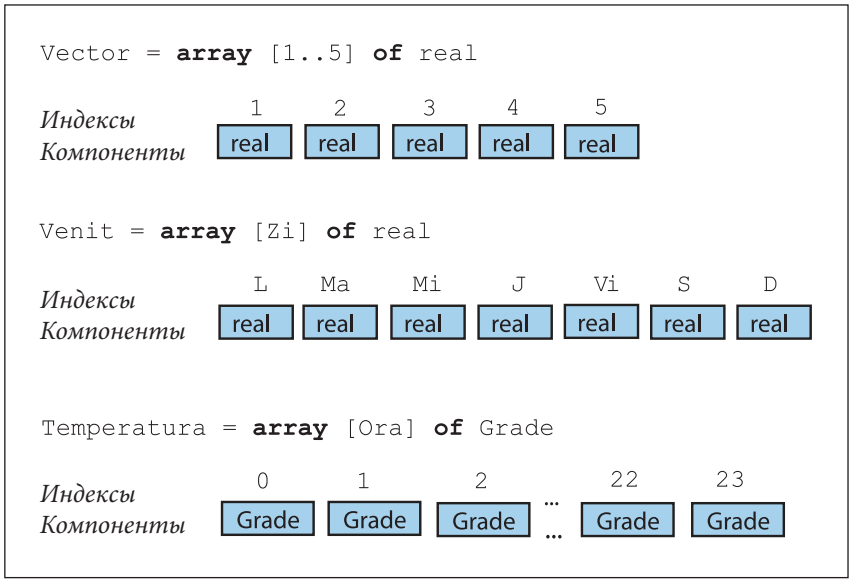


Рис. 4.1. Структура данных типа `Vector`, `Venit` и `Temperatura`

К компонентам данных типа *одномерный массив* можно применять все операции, допустимые для соответствующего базового типа. Следующая программа выводит на экран сумму компонент переменной `x` типа `Vector`. Значения переменных `x[1]`, `x[2]`, …, `x[5]` вводятся с клавиатуры.

```
Program P77;
{ Сумма компонент переменной x типа Vector }
type Vector = array [1..5] of real;
var x : Vector;
    i : integer;
    s : real;
begin
  writeln('Введите 5 чисел:');
  for i:=1 to 5 do readln(x[i]);
  writeln('Были введены:');
  for i:=1 to 5 do writeln(x[i]);
  s:=0;
  for i:=1 to 5 do s:=s+x[i];
  writeln('Сумма=', s);
  readln;
end.
```

Для того чтобы расширить область применения программы, количество компонент данных типа **`array`** рекомендуется указывать через константы.

Например, программу `P77` можно изменить таким образом, чтобы она вычисляла сумму n действительных чисел, $n \le 100$:

```
Program P78;
{ Расширение области применения программы P77 }
const nmax = 100;
type Vector = array [1..nmax] of real;
var x : Vector;
    n : 1..nmax;
    i : integer;
    s : real;
begin
  write('n='); readln(n);
  writeln('Введите  ', n, ' чисел:');
  for i:=1 to n do readln(x[i]);
  writeln('Были введены:');
  for i:=1 to n do writeln(x[i]);
  s:=0;
  for i:=1 to n do
    s:=s+x[i];
  writeln('Сумма=', s);
  readln;
end.
```

В общем виде тип *одномерный массив* определяется с помощью синтаксических диаграмм, приведенных на *рис. 4.2*. Слово **packed** (упакованный) указывает компилятору, что область памяти для элементов типа **array** должна быть выделена с применением оптимизации. Отметим, что в большинстве компьютеров использование этого префикса необязательно, так как оптимизация осуществляется автоматически.

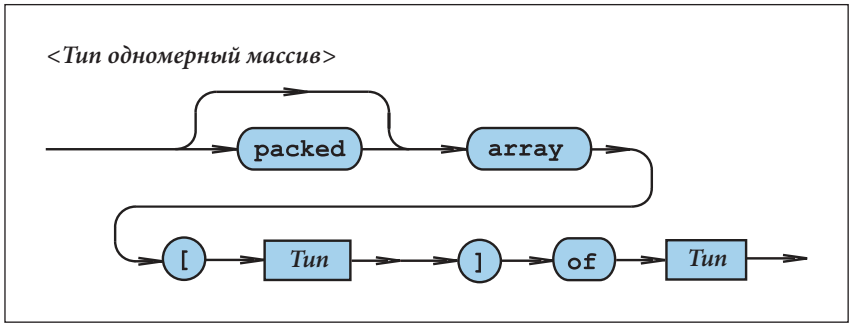


Рис. 4.2. Синтаксическая диаграмма <*Тип одномерный массив*>

Если даны две переменные типа *массив* одного и того же базового типа, то имена этих переменных могут встречаться в операциях присваивания. Такое присваивание означает копирование всех компонент массива, расположенного в правой части, в массив, расположенный в левой части.

Например, при описании

```
var a, b : Vector;
```

оператор:

```
a:=b
```

является правильным.

В примерах, рассмотренных выше, базовый тип (тип компонент) всегда был простым. Так как базовый тип в большинстве случаев может быть любым, то не исключена возможность определения массивов, компоненты которых относятся к составному типу. Рассмотрим пример, в котором базовым типом является сам тип **array**.

```
Type Linie = array [1..4] of real;
     Tabel = array [1..3] of Linie;
var L : Linie;
    T : Tabel;
    x : real;
```

Переменная T состоит из 3-х компонент: T[1], T[2] и T[3] типа Linie.
Переменная L состоит из 4-х компонент: L[1], L[2], L[3] и L[4] типа real.
Следовательно, операции присваивания

```
L[1]:=x; x:=L[3]; T[2]:=L; L:=T[1]
```

являются правильными.

Доступ к элементам переменной T может осуществляться через T[i][j] или T[i,j]. Здесь индекс i означает номер компоненты типа Linie переменной T, а j – номер компоненты типа real компоненты T[i] типа Linie.

В программах на языке ПАСКАЛЬ массивы используются для группировки под одним именем нескольких переменных, обладающих одинаковыми характеристиками.

Вопросы и упражнения

❶ Определите тип индексов и тип компонент в следующих объявлениях:

```
type P = array [1..5] of integer;
     Culoare = (Galben, Verde, Albastru, Violet);
     R = array [Culoare] of real;
     S = array [Culoare] of boolean;
     T = array [boolean] of Culoare;
```

Нарисуйте структуры данных типа P, R, S и T (*рис. 4.1*).

❷ Укажите на синтаксической диаграмме *рис. 4.2* пути, которые соответствуют объявлениям из упражнения 1.

❸ Напишите металингвистические формулы, которые соответствуют синтаксической диаграмме *<Тип массив> рис. 4.2*.

❹ Даны описания:

```
type Vector = array [1..5] of real;
var x, y : Vector;
```

Напишите арифметическое выражение, значением которого является:
а) сумма первых трех компонент переменной x;
б) сумма всех компонент переменной y;

в) произведение всех компонент переменной x;

г) абсолютное значение третьей компоненты переменной y;

д) сумма первых компонент переменных x и y.

❺ Даны описания:

```
type Zi = (L, Ma, Mi, J, Vi, S, D);
     Venit = array [Zi] of real;
var v : Venit;
```

Компоненты переменной v представляют собой ежедневный доход предприятия. Напишите программу, которая:

а) вычисляет еженедельный доход предприятия;

б) подсчитывает средний ежедневный доход;

в) определяет день, когда был получен наибольший доход;

г) определяет день, когда был получен наименьший доход.

❻ Даны описания:

```
type Ora = 0..23;
     Grade = -40..40;
     Temperatura = array [Ora] of Grade;
var t : Temperatura;
```

Компоненты переменной t представляют собой значения температуры, измеряемой каждый час в течение 24 часов. Напишите программу, которая:

а) вычисляет среднюю температуру;

б) определяет минимальное и максимальное значения температуры;

в) определяет час (часы), в который была зарегистрирована максимальная температура;

г) определяет час (часы), в который была зарегистрирована минимальная температура;

д) определяет количество дней с температурой ниже нуля;

е) определяет количество дней с температурой выше нуля.

❼ Даны описания

```
type Oras = (Chisinau, Orhei, Balti, Tighina, Tiraspol);
     Consum = array [Oras] of real;
var C : Consum;
    r : Oras;
```

Компонента C[r] переменной C представляет собой потребление электроэнергии города r в течении одной недели. Напишите программу, которая:

а) вычисляет количество электроэнергии, потребляемой всеми городами за неделю;

б) определяет город с максимальным еженедельным потреблением электроэнергии;

в) определяет город с минимальным еженедельным потреблением электроэнергии.

❽ Даны описания:

```
type Vector = array [1..5] of real;
     Matrice = array [1..3] of boolean;
     Linie = array [1..4] of real;
     Tabel = array [1..3] of Linie;
var V : Vector;
    M : Matrice;
    L : Linie;
    T : Tabel;
```

```
    x : real;
    i : integer;
```

Какие из следующих операций присваивания корректны?

a)	`T[3]:=T[1]`	*l)*	`T[2]:=V`
b)	`M:=T`	*m)*	`L:=T[3]`
c)	`L:=V`	*n)*	`T[1,2]:=M[1,2]`
d)	`L[3]:=x`	*o)*	`T[1,2]:=M[1,2]`
e)	`x:=i`	*p)*	`M[1]:=4`
f)	`i:=x`	*q)*	`M[1,3]:=L[2]`
g)	`L[3]:=i`	*r)*	`x:=T[1][2]`
h)	`i:=M[1,2]`	*s)*	`x:=M[1]`
i)	`x:=V[4]`	*t)*	`L:=M[1]`
j)	`L[3]:=V[4]`	*u)*	`V[5]:=M[3,4]`
k)	`T[1]:=4`	*v)*	`L:=M[3,4]`

❾ Используя тип данных *массив*, напишите программу, которая реализует алгоритм Эратосфена для вычисления простых чисел, меньших заданного n ($n \leq 200$).

4.2. Тип данных *строка символов*

В стандартном языке тип данных *строка символов* является частным случаем типа **`array`** и определяется конструкцией вида

```
<Имя типа> ::= packed array [1..n] of char;
```

Множеством значений данного типа являются все строки, содержащие ровно n символов.

Пример:

```
Program P80;
{ Строки символов с фиксированной длиной }
type Nume = packed array [1..8] of char;
     Prenume = packed array [1..5] of char;
var N : Nume;
    P : Prenume;
begin
  N:='Munteanu';
  P:='Mihai';
  writeln(N);
```

```
  writeln(P);
  readln;
end.
```

Результат, выводимый на экран:

```
Munteanu
Mihai
```

Так как строки различной длины принадлежат разным типам данных, в программе P80 не допустимы присваивания вида:

```
N:= 'Olaru';
P:= 'Ion';
```

В таких случаях программисту необходимо заполнить соответствующее пространство пробелами для того, чтобы в строке было ровно *n* символов, например:

```
N:= 'Olaru   ';
P:= 'Ion  ';
```

Значения любой переменной v типа **packed array** [1..n] **of** char можно ввести с клавиатуры только путем поочередного считывания соответствующих компонент:

```
read(v[1]); read(v[2]); ...; read(v[n]).
```

Однако вывести всю строку на экран можно с помощью одного вызова write(v) или writeln(v).

Особо выделим тот факт, что строки символов типа **packed array [1..n] of** char содержат ровно n символов, т. е. являются строками постоянной длины. Их длину нельзя изменять в процессе выполнения соответствующей программы. Это усложняет создание программ, предназначенных для обработки строк символов произвольной длины.

Чтобы устранить указанный недостаток, современные версии языка ПАСКАЛЬ разрешают использование строк символов произвольной длины.

В версии Turbo PASCAL тип данных *строка символов*, множеством значений которого являются строки произвольной длины, определяется с помощью конструкции вида:

type <*Имя типа*> = **string**;

или

type <*Имя типа*> = **string** [*nmax*];

где *nmax* – максимальная длина, которую могут иметь соответствующие строки. При отсутствии параметра *nmax* максимальная длина устанавливается по умолчанию, как правило – 255 символов.

К строкам типа **string** можно применять операцию конкатенации (склеивания), обозначаемую знаком «+». Текущую длину любой переменной v типа **string** можно узнать с помощью стандартной функции length(v), которая возвращает значение типа integer. Независимо от длины все строки символов типа **string** являются совместимыми.

Пример:

```
Program P81;
{ Строки символов произвольной длины }
type Nume = string [8];
     Prenume = string [5];
     NumePrenume = string;
var N : Nume;
    P : Prenume;
    NP : NumePrenume;
    L : integer;
begin
  N:='Munteanu';  L:=length(N);   writeln(N, L:4);
  P:='Mihai';     L:=length(P);   writeln(P, L:4);
  NP:=N+' '+P;    L:=length(NP);  writeln(NP, L:4);
  N:='Olaru';     L:=length(N);   writeln(N, L:4);
  P:='Ion';       L:=length(P);   writeln(P, L:4);
  NP:=N+' '+P;    L:=length(NP);  writeln(NP, L:4);
  readln;
end.
```

Результаты, выводимые на экран:

```
Munteanu   8
Mihai   5
Munteanu Mihai  14
Olaru   5
Ion   3
Olaru Ion 9
```

Отметим, что в процессе выполнения данной программы длина строк символов N, P и NP изменяется.

К строкам символов можно применять операции отношения <, <=, =, >=, >, <>. Строки сравниваются посимвольно, слева направо, в соответствии с порядковыми номерами символов типа данных **char**. Оба операнда должны относиться к типу **packed array** [1..n] **of char** с одинаковым числом компонент либо к типу **string**. Естественно, что операнды типа **string** могут быть различной длины.

Например, результатом операции:

```
'AC' < 'BA'
```

является true, а результатом операции

```
'AAAAC' < 'AAAAB'
```

является false.

Переменную типа *строка символов* можно использовать полностью или частично, обращаясь к отдельному символу строки.

Например, в строке P='Mihai' имеем P[1]='M', P[2]='i', P[3]='h' и т. д. После выполнения последовательности операторов

```
P[1]:='P';
P[2]:='e';
P[3]:='t';
P[4]:='r';
P[5]:='u';
```

переменная P примет значение 'Petru'.

Следующая программа вводит с клавиатуры произвольные строки символов и выводит на экран количество пробелов в соответствующей строке. Работа программы завершается после введения строки 'Sfîrşit'.

```
Program P82;
{ Количество пробелов в строке символов }
var S : string;
    i, j : integer;
begin
  writeln('Введите строку символов:');
  repeat
    readln(S);
    i:=0;
    for j:=1 to length(S) do
      if S[j]=' ' then i:=i+1;
    writeln('Количество пробелов=', i);
  until S='Sfîrşit';
end.
```

Вопросы и упражнения

❶ Как определяется тип данных *строка символов*?

❷ Какие операции можно применять к строкам символов?

❸ Прокомментируйте следующую программу:

```
Program P83;
{ Ошибка }
var S : packed array [1..5] of char;
begin
  S:='12345';
  writeln(S);
  S:='Sfat';
  writeln(S);
end.
```

❹ Напишите программу, которая:
а) определяет, сколько раз встречается в строке символ 'A';
б) заменяет символ 'A' символом '*';

в) удаляет из строки символ `'B'`;
г) определяет, сколько раз встречается в строке слог `'MA'`;
д) заменяет слог `'MA'` слогом `'TA'`;
е) удаляет из строки слог `'TO'`.

❺ Определите результаты операций отношения:

a) `'B' < 'A'`
b) `'BB' > 'AA'`
c) `'BAAAA' < 'AAAAA'`
d) `'CCCCD' > 'CCCCA'`
e) `'A A' = 'AA'`
f) `'BB ' < 'B B'`
g) `'A' = 'a'`
h) `'Aa' > 'aA'`
i) `'123' = '321'`
j) `'12345' > '12345'`

❻ Даны строки символов, состоящие из прописных букв латинского алфавита и пробелов. Напишите программу, которая выводит на экран данные строки согласно следующим правилам:
– все буквы от `'A'` до `'Y'` заменяются последующими буквами алфавита;
– каждая буква `'Z'` заменяется буквой `'A'`;
– все пробелы заменяются знаком `'-'`.

❼ Напишите программу, которая расшифровывает строки символов, зашифрованные согласно правилам из упражнения 6.

❽ Дано m ($m \le 100$) строк, состоящих из строчных букв латинского алфавита. Напишите программу, которая выводит на экран данные строки в алфавитном порядке.

❾ Строка S составлена из нескольких предложений, каждое из которых заканчивается точкой, восклицательным или вопросительным знаком. Напишите программу, которая выводит на экран количество предложений в данной строке.

Тест для самопроверки № 4

1. Даны следующие типы данных:

```
type Obiect = (Istoria, Geografia, Matematica,
     Informatica, Fizica);
     Nota = 1..10;
     SituatiaScolara = array [Obiect] of Nota;
```

Изобразите на рисунке структуру данных типа `SituatiaScolara`.

2. Для приведенных ниже описаний укажите тип индексов и тип компонент данных типа `OrarulLectiilor`:

```
type Lectie = 1..6;
     Obiect = (LimbaRomana, LimbaModerna, Istoria,
     Geografia, Matematica, Informatica, Fizica, Chimia);
     OrarulDeAstazi = array [Lectie] of Obiect;
```

3. Какие из приведенных ниже типов данных можно использовать в качестве индексных при описании одномерных массивов?

a) `real`

b) `integer`

c) `[-5..-1]`

d) `(AneniiNoi, Orhei, Chisinau)`

e) `boolean`

f) `char`

g) `[1..5]`

h) `array[1..5] of integer`

4. Даны следующие описания и объявления:

```
type RemunerareaLunara = array [1..12] of real;
var RMunteanu, RPetrescu : RemunerareaLunara;
    r : real;
    s : boolean;
    i : 1..12;
    t : real;
```

Какие из приведенных ниже операций присваивания являются правильными?

a) `i:=t`

b) `RMunteanu[5]:= 1461.12`

c) `RMunteanu[3]:=true`

d) `r:=RPetrescu[9]`

e) `RPetrescu[11]:=t`

f) `t:=i`

g) `RMunteanu[i]:=t`

h) `s:=RPetrescu[i]`

i) `RPetrescu:=2489,81`

j) `Petrescu[16]:=3905.00`

k) `RMunteanu:=RPetrescu`

l) `Petrescu:=2*RMunteanu`

m) `RMunteanu[s]:=r`

n) `RMunteanu[5]:=RPetrescu[12]`

o) `RMunteanu[i]:=RPetrescu[5]`

p) `s:=RMunteanu[5]=RPetrescu[8]`

5. Даны следующие описания и объявления:

```
type Tablou = array [1..10] of integer;
var x, y : Tablou;
```

Напишите арифметическое выражение для вычисления:

a) суммы первых четырех компонент переменной x;

b) суммы последних четырех компонент переменной y;

c) абсолютного значения третьей компоненты переменной x;

d) абсолютного значения шестой компоненты переменной y;

e) суммы первой компоненты переменной x и последней компоненты переменной y.

6. Даны n ($n \le 50$) целых чисел $a_1, a_2, a_3, ..., a_n$. Напишите программу на ПАСКАЛЕ, которая вводит с клавиатуры рассматриваемые числа и выводит их на экран в порядке, обратном вводу: $a_n, ..., a_3, a_2, a_1$.

7. Рассматриваются одномерные массивы A`[1..100]` и B`[1..100]`, индексы и компоненты которых являются натуральными числами. Напишите программу на ПАСКАЛЕ,

которая вычисляет число случаев, для которых компоненты с одним и тем же индексом из рассматриваемых массивов равны, то есть A[i]=B[i].

8. Какие операции могут быть выполнены над строками символов? Укажите тип результата этих операций.

9. Напишите программу на ПАСКАЛЕ, которая выводит на экран строку символов в порядке, обратном тому, в котором она была введена с клавиатуры. Например, строка 'soare' будет выведена на экран как 'eraos'.

10. Даны строки символов, состоящие из заглавных букв латинского алфавита. Напишите программу, которая выводит на экран количество гласных букв в считанной с клавиатуры строке символов S .

11. Напишите программу на ПАСКАЛЕ, которая вводит с клавиатуры натуральное число из интервала [1, 7] и выводит на экран название соответствующего дня недели.

Примеры:

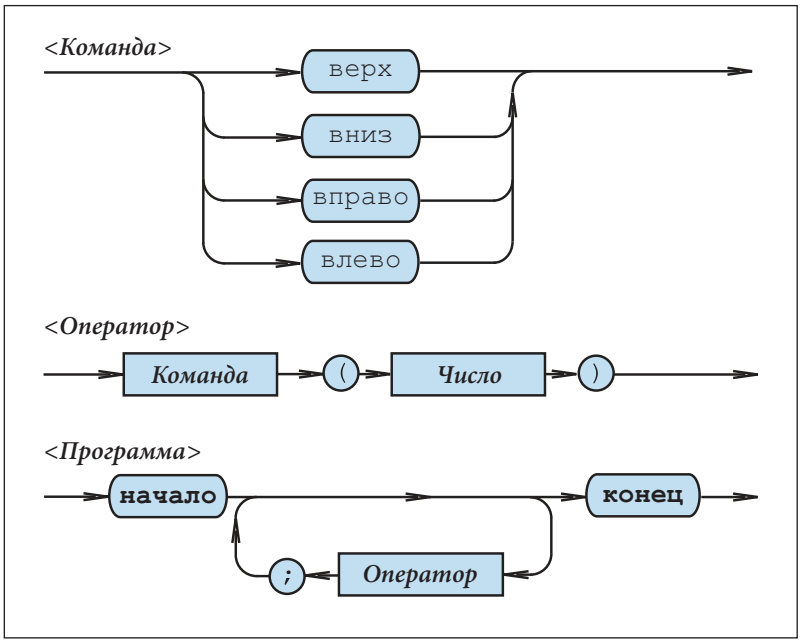
	Вход	*Выход*
a)	3	Miercuri
b)	1	Luni
c)	6	Simbata

Ответы на задания из тестов для самопроверки

Тест № 1

1. *a, c, d* – верно; *b, e, f, g* – ошибочно.

2.



3. *a, c, e, f , h, i, m, o* – верно; *b, d, g, j, k, l, n* – ошибочно.

4. *<Восьмеричная цифра>* ::= `0 | 1 | 2 | 3 | 4 | 5 | 6 | 7`

<Восьмеричное число> ::= [+|-]*<Восьмеричная цифра>*{*<Восьмеричная цифра>*}

5. *<Цифра>* ::= `0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9`

<Буква> ::= `a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z`

<Идентификатор> ::= *<Буква>* { *<Буква>* | *<Цифра>* }

6.

1) `x1`

2) `x2`

3) `Delta`

4) `UnghiulAlfa`

5) `UnghiulBeta`

6) `DistantaParcursa`

7)	`ListaElevilor`	*9)*	`Pixel`
8)	`Vocala`	*10)*	`Culoare`

7.

a)	`3.14`	*f)*	`-984.52`	*k)*	`-3628.297e12`
b)	`265`	*g)*	`-523`	*l)*	`-38.00001`
c)	`23.4635`	*h)*	`+28`	*m)*	`35728.345452e-8`
d)	`+0.000001`	*i)*	`+28000000`	*n)*	`24815`
e)	`6.1532e-5`	*j)*	`614.45e-12`	*o)*	`-296.0020001`

8.

a)	6124,485;	*f)*	$-0{,}03428 \cdot 10^{-8}$;	*k)*	2005;
b)	+18,315;	*g)*	232847,5213;	*l)*	$+23{,}08 \cdot 10^{-5}$;
c)	$-218{,}034 \cdot 10^{-3}$;	*h)*	$-0000012 \cdot 10^{+2}$;	*m)*	–17502;
d)	193526;	*i)*	18,45;	*n)*	+1;
e)	$1000{,}01 \cdot 10^{23}$;	*j)*	$623{,}495 \cdot 10^{-6}$;	*o)*	$-46341{,}2 \cdot 10^{-6}$.

9. Ключевые слова: **`Program`**, **`var`**, **`begin`**, **`if`**, **`then`**, **`end`**, **`and`**.
Специальные символы: `;` , `,` , `:` , `(` , `)` , `<>` , `:=` , `/` , `'` , `=` , `..`
Идентификаторы: TA1, a, b, x, `real`, `readln`, `writeln`.
Числа: `0`.
Строки символов:
```
'Уравнение имеет единственный корень',
'Уравнение имеет бесконечное множество корней',
'Множество корней уравнения пусто'.
```

10. *Строка 1* – заголовок; *Строка 2* – раздел описаний; *Строки 3–15* – раздел операторов.

Тест № 2

1. Под типом данных понимается множество значений и множество операций, которые могут выполняться с соответствующими значениями. Примеры типов данных: `integer`, `real`, `char`. Значения `1`, `2` и `3` – типа `integer`; Значения `1.0`, `2.0` и `0.5e+07` – типа `real`, а значения `'A'`, `'B'` и `'+'` – типа `char`.

2. В программах на языке ПАСКАЛЬ величины являются объектами, предназначенными для представления данных. Существуют величины двух видов: переменные и постоянные. Значение любой переменной во время исполнения программы может быть изменено, в то время как значения постоянных величин не могут быть модифицированы.

3. `i`, `j` — переменные типа `integer`; a, b, c — переменные типа `real`; s — переменная типа `char`; p — переменная типа `boolean`; 5, 9 — константы типа `integer`; `1.0`, `1.0e-`

01, -2.001 — константы типа real; 'A' — константа типа char; true — константа типа boolean.

4. Множество значений типа данных integer состоит из целых чисел, которые могут быть представлены в компьютере, на котором реализован язык программирования ПАСКАЛЬ. Наибольшее допустимое значение можно узнать с помощью константы MaxInt, доступной в любой программе ПАСКАЛЬ. Обычно, наименьшее допустимое целое число данного типа равно -MaxInt или –(MaxInt+1). Примеры операций, допустимых над целыми значениями: +, -, *, **mod**, **div** и др.

5. Ошибки переполнения возникают в случае, если x*y>MaxInt. Например, для MaxInt=32767 (версия Turbo PASCAL 7.0), ошибки переполнения возникают, если пользователь вводит для x и y значения больше, чем 200.

6. Множество значений типа данных real состоит из вещественных чисел, которые могут быть представлены в компьютере, на котором реализован язык программирования ПАСКАЛЬ. Примеры операций, допустимых над вещественными значениями: +, -, *, / (деление) и др. Результаты данных операций являются приближенными вследствие ошибок округления.

7. Ошибки переполнения появятся в случае, когда результат операции x*y выходит за область значений типа данных real. В версии Turbo PASCAL 7.0 областью значений типа real является $-1{,}7\cdot10^{38}$, ..., $+1{,}7\cdot10^{38}$. Следовательно, ошибки переполнения возникают в случае, если пользователь вводит для x и y значения больше, например, чем 10^{20}.

8. Тип данных boolean включает значения истинности false и true. Предопределенными операциями типа данных boolean являются логические операции **not**, **and** и **or**.

9. См. *рис. 2.1*.

10. Оператор readln(p, q) ошибочен вследствие того, что значения переменных логического типа не могут быть считаны с клавиатуры.

11. Множество значений типа char включает все печатные символы, упорядоченные согласно таблице кодов *ASCII*. Примеры операций допустимые над значениями типа char: ord, pred, succ, операции отношения.

12.

```
Program RTA1;
 { Порядковые номера десятичных цифр }
begin
  writeln(ord('0'));
  writeln(ord('1'));
  writeln(ord('2'));
  writeln(ord('3'));
  writeln(ord('4'));
  writeln(ord('5'));
  writeln(ord('6'));
  writeln(ord('7'));
  writeln(ord('8'));
  writeln(ord('9'));
end.
```

13. Множество значений перечисляемого типа данных задается списком идентификаторов. Первый в списке идентификатор задает минимальное значение, его порядковый номер равен нулю. Второй идентификатор определяет значение с порядковым номером один, третий – с номером два и т.д. Примеры операций допустимые над значениями перечисляемого типа: `ord`, `pred`, `succ`, операции отношения.

14.

```
Program RTA2;
 { Порядковые номера значений перечисляемого типа }
type FunctiaOcupata = ( Muncitor, SefDeEchipa, Maistru,
                        SefDeSantier, Director);
     StareaCivila = (Casatorit, Necasatorit);
begin
  writeln(ord(Muncitor));
  writeln(ord(SefDeEchipa));
  writeln(ord(Maistru));
  writeln(ord(SefDeSantier));
  writeln(ord(Director));
  writeln(ord(Casatorit));
  writeln(ord(Necasatorit));
end.
```

15. Интервальный тип определяет подмножество значений некоторого, уже определенного типа: `integer`, `boolean`, `char` или *перечисляемого*, называемого базовым типом. Над значениями интервального типа допустимы все операции базового типа.

16. p – интервальный тип, базовый тип `char`. Может принимать значения *'A', 'B', 'C', ..., 'Z'*;

q – интервальный тип, базовый тип `integer`. Может принимать значения 1, 2, 3, ..., 9;

r – интервальный тип, базовый тип `char`. Может принимать значения *'0', '1', '2', ..., '9'*;

s – тип `char`. В качестве значения может принимать любой печатный символ *ASCII*;

t – тип `integer`. В качестве значения может принимать любое целое число, допустимое в конкретной машинной реализации языка;

u – перечисляемый тип. Может принимать значения `Alfa, Beta, Gama, Delta`.

17. Порядковые типы данных: `integer`, `boolean`, `char`, *перечисляемый*, *интервальный*. Общие свойства:

a) любое значение порядкового типа имеет порядковый номер, который можно узнать с помощью предопределенной функции `ord`;

b) к значениям любого порядкового типа можно применить операции отношения;

c) для порядковых типов существуют предопределенные функции `pred` (предшествующий) и `succ` (последующий).

18.

```
Y    E    -6    10    1    3
```

19.

```
1    0    2    true    false    true
```

20. Два типа являются идентичными, если они описаны тем же именем.
Два типа являются совместимыми в случае истинности одного из следующих утверждений:
a) рассматриваемые типы идентичны;
b) один тип является интервальным типом второго;
c) оба типа являются интервальными, с одним и тем же базовым типом.
Тип данных является анонимным, если он описан неявно при объявлении переменных.

21. Идентичные типы: *a)* `T1`, `integer` и `T2`; *b)* `T3` и `T4`;
Совместимые типы: *a)* `T1`, `integer`, `T2`, `T3`, `T4`, `T5` и `1..100`; *b)* `T6`, `T7` и `T8`; *c)* `T9` и `T10`.
Анонимные типы: *a)* `1..100`; *b)* `(Alfa, Beta, Gama, Delta)`.

22. `Alfa` – `integer`; `Beta` — `real`; `Indicator` – `boolean`; `Mesaj` – строка символов; `Semn` – `char`; `Inscris` – строка символов.

23.

```
const  a = 3.14;
       b = 9.8;
var  i, j : integer;
     x, y : real;
```

Тест № 3

1.

a) `(a+b)-2*a*b`

b) `6*sqr(a)+15*a*b-13*sqr(b)`

c) `(a+b)*(a-b)`

d) `2*Alfa*Beta-5*Pi*r`

e) `Pi*sqr(r)+Alfa*sqr(Beta)`

f) `x and y or x and z`

2.

a) $a^2+\frac{2}{b^2};$

b) $\frac{2a}{b+c};$

c) $15\sqrt{\frac{a}{a-b}};$

d) $\overline{xy}\vee z;$

e) $\left(\frac{a+b}{2}\right)^2;$

f) $(x\neq 0)\ \&\ (q<p).$

3. *a*, *c*, *e*, *f* – правильно; *b*, *d* – ошибочно.

4. *a*) 14; *b*) 6; *c*) –4; *d*) `false`; *e*) `true`; *f)* `true`.

5.

a) `integer`

b) `real`

c) `real`

d) `integer`

e) `integer`

f) `integer`

g) `boolean`

h) `char`

6.

```
Program RTA3;
var  i : integer;
     x, y : real;
begin
  writeln('Введите i='); readln(i);
  writeln('Введите x='); readln(x);
  writeln('Введите y='); readln(y);
  writeln(15*i*(x+y));
  readln;
end.
```

7. *a*, *c*, *e*, *f* – верно; *b*, *d* – ошибочно.

8.

```
Program RTA4;
var x, y : real;
begin
  write('x='); readln(x);
  if x>15  then y:=9*x+3*sqr(x)
           else y:=3*x-5*sqrt(x+28);
  writeln('y=', y);
  readln;
end.
```

9.

```
Program RTA5;
var i : integer;
begin
  write('Введите числовое значение монеты: ');
  readln(i);
  case i of
    1 : writeln('один бань');
    5 : writeln('пять бань');
    10 : writeln('десять бань');
    25 : writeln('двадцать пять бань');
    50 : writeln('пятьдесят бань');
```

```
    else writeln('недопустимое значение');
  end;
  readln;
end.
```

10.

```
Program RTA6;
var  n, i : integer;
     s, p : real;
begin
  write('n='); readln(n);
  s:=0; p:=1;
  for i:= 1 to n do
    begin
      s:=s+(1/i); p:=p*(1/i);
    end;
  writeln('s=', s);
  writeln('p=', p);
  readln;
end.
```

11.

```
Program RTA7;
var y, x, x1, x2, deltaX : real;
begin
  write('x1='); readln(x1);
  write('x2='); readln(x2);
  write('deltaX='); readln(deltaX);
  writeln('x':10, 'y':20);
  writeln;
  x:=x1;
  while x<=x2 do
    begin
      if x>=4 then y:=2*sqrt(x+6) else y:=3-abs(x);
      writeln(x:20, y:20);
      x:=x+deltaX;
    end;
  readln;
end.
```

12.

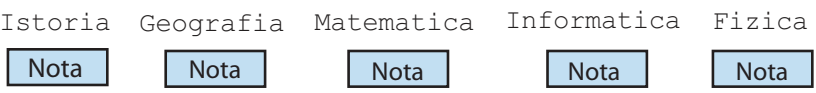
```
Program RTA8;
var  c : char; { символ считываемый с клавиатуры }
     n : integer; { количество цифр в сообщении }
```

```
begin
  n:=0;
  writeln('Введите сообщение:');
  repeat
    readln(c);
    if (c<>'*') and (c<>'#') then n:=n+1;
  until c='#';
  writeln('Количество цифр n=', n);
readln;
end.
```

Тест № 4

1.

```
type SituatiaScolara = array [Obiect] of Nota
```



2. Тип индекса – `Lectie`; тип компонент – `Obiect`.

3. В качестве индексного типа можно использовать (*b*), (*c*), (*d*), (*e*), (*f*), (g).

4. Правильными являются операции присваивания (*b*), (*d*), (*e*), (*f*), (*g*), (*k*), (*n*), (*o*), (*p*).

5.

a) `x[1]+ x[2]+ x[3]+ x[4]`

b) `y[7]+ y[8]+ y[9]+ y[10]`

c) `abs(x[3])`

d) `abs(x[6])`

e) `x[1]+ y[10]`

6.

```
Program RTA9;
{  Ответ на Тест 4, задание 6 }
type Numere = array [1..50] of integer;
var A : Numere;
    n, i : integer;
begin
  write('Введите n='); readln(n);
  { Считываем числа с клавиатуры }
  for i:=1 to n do
    begin
      write('A[', i, ']='); readln(A[i]);
    end;
```

```
  { Выводим числа на экран }
  writeln('Числа в обратном порядке:');
  for i:=n downto 1 do
    writeln('A[', i, ']=', A[i]);
  readln;
end.
```

7.

```
Program RTA10;
{ Ответ на Тест 4, задание 7 }
type Natural = 0..MaxInt;
     Tablou = array [1..100] of Natural;
var A, B : Tablou;
    n : 1..100; { текущее число компонент }
    c : 0..100; { число случаев A[i]=B[i] }
    i : 1..100; { индекс }
begin
   write('Введите число компонент n=');
   readln(n);
   writeln('Введите компоненты массива A');
   for i:=1 to n do
     begin write('A[', i, ']='); readln(A[i]); end;
   writeln('Введите компоненты массива B');
   for i:=1 to n do
      begin write('B[', i, ']='); readln(B[i]); end;
   c:=0;
   for i:=1 to n do
     if A[i]=B[i] then c:=c+1;
   writeln('Число случаев c=', c);
   readln;
end.
```

8. Над строками типа **string** можно выполнять операцию конкатенации (сцепления, слияния), обозначаемую как „+". Текущая длина любой величины v типа **string** может быть найдена с помощью предопределенной функции `length(v)`, возвращающей значение типа `integer`. Над строками типа **string** разрешены операции отношения `<, <=, =, >=, >, <>`. Результат этих операций имеет тип `boolean`.

9.

```
Program RTA11;
{ Ответ на Тест 4, задание 9 }
var S : string;
    i : integer;
begin
  writeln('Введите строку:');
  readln(S);
```

```
  writeln('Строка в обратном порядке:');
  for i:=length(S) downto 1 do
    write(S[i]);
    writeln;
  readln;
end.
```

10.

```
Program RTA12;
{ Ответ на Тест 4, задание 10 }
var S : string;
    i, n : integer;
begin
  write('Введите строку символов, состоящую из ');
  writeln('заглавных букв латинского алфавита:');
  readln(S);
  { Подсчитываем гласные в строке }
  n:=0;
  for i:=1 to length(S) do
    if (S[i]='A') or (S[i]='E') or (S[i]='I') or
   (S[i]='O') or (S[i]='U') then n:=n+1;
  writeln('Количество гласных = ', n);
readln;
end.
```

11.

```
Program RTA13;
{ Ответ на Тест 4, задание 11 }
type ZileleSaptaminii = array [1..7] of string;
var Z : ZileleSaptaminii;
    i : 1..7;
begin
   Z[1]:='Luni';
   Z[2]:='Marti';
   Z[3]:='Miercuri';
   Z[4]:='Joi';
   Z[5]:='Vineri';
   Z[6]:='Simbata';
   Z[7]:='Duminica';
   write('Введите номер дня недели i=');
   readln(i);
   writeln(Z[i]);
   readln;
end.
```

Приложение 1. Словарь языка ПАСКАЛЬ

1. <*Буква*> ::= a|b|c|d|e|f|g|h|i|j|k|l|m|n|o|p|q|r|s|t|u|v|w|x|y|z

2. <*Цифра*> ::= 0|1|2|3|4|5|6|7|8|9

3. <*Специальный символ*> ::= +|-|*|/|=|<|>|]|[|,|(|)|:|;|^|.|@|{|}|$|#|<=
|>=| <> | := |..| <*Ключевое слово*> | <*Эквивалентный символ*>

4. <*Эквивалентный символ*> ::= (*|*)|(.|.)

5. <*Ключевое слово*> ::= **and | array | begin | case| const | div | do | downto | else | end | file | for | function | goto | if | in | label | mod | nil | not | of | or | packed | procedure | program | record | repeat | set | then | to | type | until | var | while |with**

6. <*Идентификатор*> ::= <*Буква*> { <*Буква*> | <*Цифра*> }

7. <*Директива*> ::= <*Буква*> {<*Буква*> | <*Цифра*>}

8. <*Целое без знака*> ::= <*Цифра*> {<*Цифра*>}

9. <*Знак*> ::= + | –

10. <*Целое число*> ::= [<*Знак*>] <*Целое без знака*>

11. <*Масштабный множитель*> ::= <*Целое число*>

12. <*Вещественное число*> ::= <*Целое число*> e <*Масштабный множитель*> |
<*Целое число*> . <*Целое без знака*> [e <*Масштабный множитель*>]

13. <*Число*> ::= <*Целое число*> | <*Вещественное число*>

14. <*Строка*> ::= ' <*Элемент строки*> { <*Элемент строки*> } '

15. <*Элемент строки*> ::= '' | <*Любой печатный символ*>

16. <*Метка*> ::= <*Целое без знака*>

17. <*Комментарий*> ::= (* <*Любая последовательность символов, символов конца строки и возврата каретки за исключением фигурных скобок*> *)

Примечание. Терминальные символы { и } из формулы (17) представлены через эквивалентные символы (* и *) соответственно.

Приложение 2. Синтаксис языка ПАСКАЛЬ

1. <*Программа*> ::= <*Заголовок программы*> <*Блок*>**.**

2. <*Заголовок программы*> ::= **Program** <*Идентификатор*> [**(**<*Идентификатор*> {**,** <*Идентификатор*>}**)**]**;**

3. <*Блок*> ::= <*Раздел описаний*>
<*Раздел операторов*>

4. <*Раздел описаний*> ::= [<*Раздел описания меток*>]
[<*Раздел описания констант*>]
[<*Раздел описания типов*>]
[<*Раздел описания переменных*>]
[<*Раздел описания подпрограмм*>]

5. <*Раздел описания меток*> ::= **label** <*Метка*> {**,** <*Метка*>}**;**

6. <*Раздел описания констант*> ::= **const** <*Описание константы*>**;** {<*Описание константы*>**;**}

7. <*Описание константы*> ::= <*Идентификатор*> **=** <*Константа*>

8.<*Константа*> ::= [**+** | **–**] <*Число без знака*>| [**+** | **–**]<*Имя константы*>| <*Строка символов*>

9. <*Раздел описания типов*> ::= **type** <*Описание типа*>**;** {<*Описание типа*>**;**}

10. <*Описание типа*>::= <*Идентификатор*> **=** <*Тип*>

11. <*Раздел описания переменных*> ::= **var** <*Описание переменной*>**;**{<*Описание переменной*>**;**}

12. <*Описание переменной*> ::= <*Идентификатор*> {**,**<*Идентификатор*>} **:** <*Тип*>

13. <*Раздел описания подпрограмм*> :: = {< *Функция* >**;** | < *Процедура* >**;**}

14. <*Тип*> ::= <*Идентификатор*> |
<*Перечисляемый тип*> |
<*Интервальный тип*> |
<*Тип-массив*> |
<*Тип-запись*> |
<*Тип-множество*> |
<*Файловый тип*> |
<*Ссылочный тип*>

15. <*Перечисляемый тип*> ::= **(**<*Идентификатор*> {**,**<*Идентификатор*>}**)**

16. <*Интервальный тип*> ::=<*Константа*> **. .** <*Константа*>

17. <*Тип массив*> ::= [**packed**] **array** **(** **.**<*Тип*> {**,**<*Тип*>}**.** **)** **of** <*Тип*>

18. <*Тип множество*> ::= **[packed] set of** <*Тип*>

19. <*Файловый тип*> ::= **[packed] file of** <*Тип*>

20. <*Ссылочный тип*> ::= **^**<*Тип*>

21. <*Тип запись*> ::= [**packed**] **record** <*Список полей*> [**;**] **end**

22. <*Список полей*> ::= <*Фиксированная часть*> [**;** <*Вариантная часть*>] | <*Вариантная часть*>

23. <*Фиксированная часть*> ::= <*Секция записи*> {**;** <*Секция записи*>}

24. <*Секция записи*> ::= <*Имя поля*> {**,** <*Имя поля*>} **:** <*Тип*>

25. <*Вариантная часть*> ::= **case** [<*Идентификатор*>**:**] <*Тип*> **of** <*Вариант записи*> {**;** <*Вариант записи*>}

26. <*Вариант записи*> ::= <*Константа*> {**,**<*Константа*>} **:** ([<*Список полей*>][**;**])

27. <*Функция*> :: = <*Заголовок функции*>**;**<*Блок*> | <*Заголовок функции*>**;** <*Директива*>| **function** <*Идентификатор*>**;**<*Блок*>

28. <*Заголовок функции*> ::= **function** <*Идентификатор*> [<*Список формальных параметров*>] **:** <*Идентификатор*>

29. <*Процедура*> ::= <*Заголовок процедуры*>**;** <*Блок*>| <*Заголовок процедуры*>**;** <*Директива*>| **procedure** <*Идентификатор*>**;** <*Блок*>

30. <*Заголовок процедуры*> : = **procedure** <*Идентификатор*> [<*Список формальных параметров*>]

31. <*Список формальных параметров*> ::= **(**<*Формальный параметр*> {**;** <*Формальный параметр*>}**)**

32. <*Формальный параметр*> ::= [**var**] <*Идентификатор*> {**,** <*Идентификатор*>} **:** <*Идентификатор*> | <*Заголовок функции*> | <*Заголовок процедуры*>

33. <*Оператор*> ::= [<*Метка*>**:**] <*Оператор без метки*>

34. <*Оператор без метки*> ::= <*Оператор присваивания*> | <*Оператор процедуры*> |
<*Составной оператор*> | <*Оператор* **if**> |
<*Оператор* **case**> | <*Оператор* **while**> |
<*Оператор* **repeat**> | <*Оператор* **for**> |
<*Оператор* **with**> | <*Оператор* **goto**> |
<*Пустой оператор*>

35. <*Оператор присваивания*> ::= <*Переменная*> **:=** <*Выражение*> | <*Имя функции*> **:=** <*Выражение*>

36. <*Оператор процедуры*> :: = <*Имя процедуры*> [<*Список фактических параметров*> | <*Список параметров вывода*>]

37. <*Список фактических параметров*> ::= (<*Фактический параметр*> {**,** <*Фактический параметр*>})

38. <*Фактический параметр*> ::= <*Выражение*> |<*Переменная*>| <*Имя функции*> | <*Имя процедуры*>

39. <*Имя функции*> ::= <*Идентификатор*>

40. <*Имя процедуры*> ::= <*Идентификатор*>

41. <*Список параметров вывода*> ::= (<*Параметр вывода*> {**,** <*Параметр вывода*>})

42. <*Параметр вывода*> ::= <*Выражение*> [**:** <*Выражение*> [**:** <*Выражение*>]]

43. <*Составной оператор*>::= **begin** <*Оператор*> {**;** <*Оператор*>} **end**

44. <*Оператор* **if**>::= **if** <*Логическое выражение*> **then** <*Оператор*> [**else** <*Оператор*>]

45. <*Оператор* **case**> ::= **case** <*Выражение*> **of** [<*Вариант*>{**;** < *Вариант* >}][**;**] **end**

46. < *Вариант* > ::= <*Константа*> {**,** <*Константа*>} **:** <*Оператор*>

47. <*Оператор* **while**>::= **while** <*Логическое выражение*> **do** <*Оператор*>

48. <*Оператор* **repeat**> ::= **repeat** <*Оператор*> {**;**<*Оператор*>} **until** <*Логическое выражение*>

49. <*Логическое выражение*> ::= <*Выражение*>

50. <*Оператор* **for**> ::= **for** <*Переменная*> **:=** <*Выражение*> <*Шаг*> <*Выражение*> **do** <*Оператор*>

51. <*Шаг*> ::= **to** | **downto**

52. <*Оператор* **with**> ::= **with** <*Переменная*> {**,**<*Переменная*>} **do** <*Оператор*>

53. <*Оператор* **goto**> ::= **goto** <*Метка*>

54. <*Пустой оператор*> ::=

55. <*Переменная*> ::= <*Идентификатор*> | <*Переменная*> **(.** <*Выражение*> {**,** <*Выражение*>}**.)** | <*Переменная*> **.** <*Имя поля*> | <*Переменная*>**^**

56. <*Имя поля*> ::= <*Идентификатор*>

57. <*Выражение*> ::= <*Простое выражение*> <*Операция отношения*> <*Простое выражение*>

58. <*Операция отношения*> ::= **<** | **<=** | **=** | **>=** | **>** | **<>** | **in**

59. <*Простое выражение*> ::= **[+|-]** <*Терм*> {<*Аддитивная операция*> <*Терм*>}

60. <*Аддитивная операция*> ::= **+|-|or**

61. <*Терм*> ::= <*Фактор*> {<*Мультипликативная операция*> <*Фактор*>}

62. <*Мультипликативная операция*> ::= ***** | **/** | **div** | **mod** | **and**

63. <*Фактор*> ::= <*Переменная*> | <*Константа без знака*> | <*Вызов функции*> | **not** <*Фактор*> | (<*Выражение*>) | <*Конструктор множества*>

64. <*Вызов функции*> ::= <*Имя функции*> [<*Список фактических параметров*>]

65. <*Константа без знака*> ::= <*Число без знака*> | <*Строка символов*> | <*Идентификатор*> | **nil**

66. <*Конструктор множества*> ::= **(.**[<*Описание элемента*> {**,** <*Описание элемента*>}]**.)**

67. <*Описание элемента*> ::= <*Выражение*> [**..** <*Выражение*>]

68. <*Раздел операторов*> ::= <*Составной оператор*>

Примечание. Терминальные символы [и] из формул (17), (55) и (66) представлены через эквивалентные символы (. и .) соответственно.

Приложение 3. Компиляция и отладка ПАСКАЛЬ программ

После написания, программы на языке ПАСКАЛЬ необходимо отредактировать, скомпилировать и отладить.

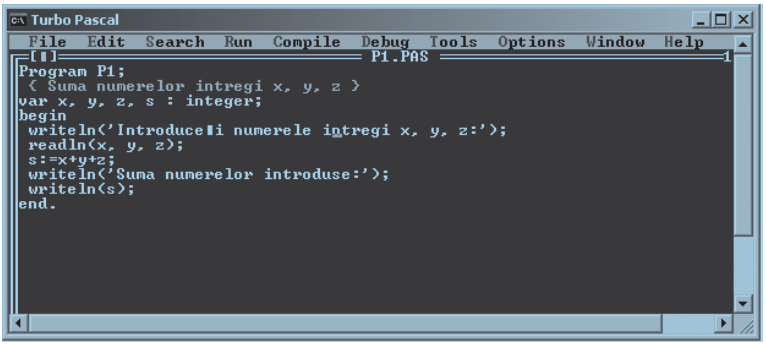
Редактирование состоит в вводе программы в компьютер и выявлении возможных ошибок, возникающих при наборе текста с клавиатуры. Введенные программы можно сохранять в файлах с расширением „`.pas`".

Компиляция заключается в автоматическом переводе ПАСКАЛЬ программы на машинный язык. После компиляции, программа на машинном языке может быть сразу запущена на выполнение или сохранена в выполняемом файле с расширением „`.exe`".

Отладка состоит в выявлении и устранении синтаксических и семантических ошибок.

Как правило, перечисленные выше операции выполняются с помощью специальной компьютерной программы, называемой *интегрированная среда для разработки* (IDE – *Integrated Development Environment*).

При использовании интегрированных сред для разработки ПАСКАЛЬ программ, взаимодействие человек-компьютер осуществляется с помощью графических интерфейсов, которые отображают на экране монитора окна приложений, диалоговые окна, окна для навигации, окна документов.



В общем случае, окна интегрированных сред для разработки программ содержат стандартные элементы, которые были изучены в предыдущих классах: строки меню, меню, команды, кнопки, курсоры, кассеты и т.д.

Ниже приведены часто используемые команды из меню сред для разработки программ Turbo PASCAL и Free PASCAL, которые установлены на большинстве школьных компьютеров нашей страны.

Меню `File` (Файл)

`New` (**Новый**) – создается новое окно, в котором пользователь может вводить и редактировать текст, обычно, программу на ПАСКАЛЕ.

Open... (**Открыть...**) – читается файл, указанный пользователем, и его содержимое отображается в новом окне. Далее, пользователь может редактировать, компилировать и отлаживать содержимое рассматриваемого окна.

Save (**Сохранить**) – содержимое текущего окна сохраняется в открытом ранее с помощью команды **Open** файле. Если же текущее окно было открыто с помощью команды New, будет создан новый файл, имя которого пользователь должен ввести с клавиатуры.

Save as... (**Сохранить как...**) – содержимое текущего окна сохраняется в новом файле. Пользователь должен ввести имя вновь создаваемого файла.

Print (**Печатать**) – содержимое текущего окна выводится на печать.

Exit (**Выход**) – выход из среды для разработки программ. Перед выходом, пользователю будет предложено сохранить содержимое всех открытых окон.

Меню Edit (Редактирование)

Cut (**Вырезать**) – выделенный фрагмент удаляется из редактируемого текста. Вырезанный фрагмент запоминается в буферной памяти.

Copy (**Скопировать**) – выделенный фрагмент текста копируется в буферную память.

Paste (**Вставить**) – содержимое буферной памяти вставляется в редактируемый текст, начиная с места, где находится курсор.

Clear (**Очистить**) – удаляет выделенный фрагмент текста без его запоминания в буферной памяти.

Меню Search (Поиск)

Find (**Искать**) – ищет в редактируемом тексте, начиная с места, где находится курсор, заданный фрагмент текста.

Replace (**Заменить**) – заменяет выделенный фрагмент редактируемого текста на текст заданный пользователем.

Меню Run (Пуск)

Run (**Пуск**) – компилирует и запускает на выполнение ПАСКАЛЬ программу из текущего окна. Если программа содержит ситаксические ошибки, выводится сообщение с указанием вида и места ошибки.

Меню Compile (Компиляция)

Compile (**Скомпилировать**) – компилирует ПАСКАЛЬ программу из текущего окна, без ее запуска на выполнение.

Build (**Создать**) – компилирует ПАСКАЛЬ программу из текущего окна и сохраняет результат компиляции в виде выполняемого файла.

Меню Windows (**Окна**) содержит команды предназначенные для размещения окон на экране и переключения между ними, а меню Help (**Справка**) – команды, представляющие доступ к справочной информации.

Acest manual este proprietatea Ministerului Educației al Republicii Moldova

Gimnaziul/Liceul _____
Manualul nr. _____

Anul de folosire	Numele de familie și prenumele elevului	Anul școlar	Aspectul manualului	
			la primire	la restituire
1.				
2.				
3.				
4.				
5.				

- Dirigintele trebuie să controleze dacă numele elevului este scris corect.
- Elevul nu trebuie să facă niciun fel de însemnări în manual.
- Aspectul manualului (la primire și la restituire) se va aprecia folosind termenii: *nou, bun, satisfăcător, nesatisfăcător*.

Imprimare la Tipografia „BALACRON" SRL,
str. Calea Ieșilor, 10;
MD-2069, Chișinău, Republica Moldova
Comanda nr. 663

9 789975 850148