§ 14. Язык программирования Паскаль

Компьютер (от англ. *computer* — вычислитель) — устройство или система, способные выполнять заданную четко определенную изменяемую последовательность операций (чаще всего численных расчетов).

Электронно-вычислительная машина (ЭВМ) — комплекс технических средств, где основные функциональные элементы (логические, запоминающие, индикационные и др.) выполнены на электронных приборах, предназначенных для автоматической обработки информации в процессе решения вычислительных задач.

Никлаус Вирт (родился в 1934 г.) — швейцарский ученый, специалист по информатике, один из известнейших теоретиков в области разработки языков программирования, профессор компьютерных наук. Создатель и ведущий проектировщик языков программирования Паскаль, Модула-2, Оберон.

Желание упростить и ускорить всевозможные расчеты присуще человеку с древних времен. Сегодня компьютер способен выполнять сотни миллионов операций в секунду. Для решения вычислительных задач требуется сначала составить алгоритм их решения, а затем записать его в виде программы, используя какой-либо язык программирования.

Язык программирования устанавливает набор правил, определяющих внешний вид программы и действия, которые выполнит исполнитель под ее управлением.

Язык программирования Паскаль (Pascal) используется для обучения программированию и является базой для ряда профессиональных языков программирования.

Существует множество сред программирования, поддерживающих язык Паскаль: PascalABC, FreePascal, Delphi, GNU Pascal, Dev-Pascal, Rad Studio и др. В учебном курсе используется среда PascalABC (с ней вы работали, знакомясь с учебными компьютерными исполнителями).

14.1. Команда вывода

Демонстрировать работу любой программы имеет смысл только тогда, когда она выводит какую-либо информацию.

Программа на языке Pascal (тело программы) должна начинаться со слова **begin**, а заканчиваться словом **end** и точкой. Программа, состоящая из этих команд, разделенных пробелом или переводом строки, может быть запущена на выполнение, но она ничего не делает. Добавим в нее команду вывода приветствия:

```
begin
 write('Привет!');
end.
```

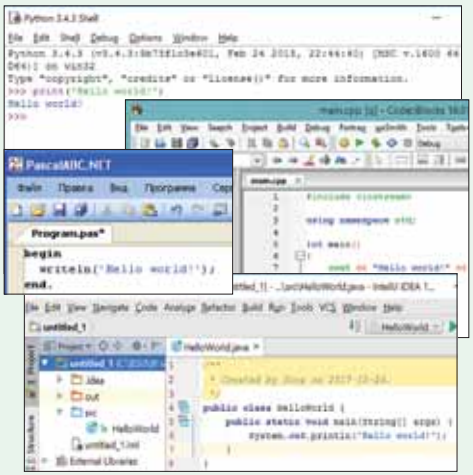
Результат работы программы отражается в нижней части окна программы PascalABC в окне вывода (пример 14.1).

Команда `write( );` предназначена для **вывода данных.**

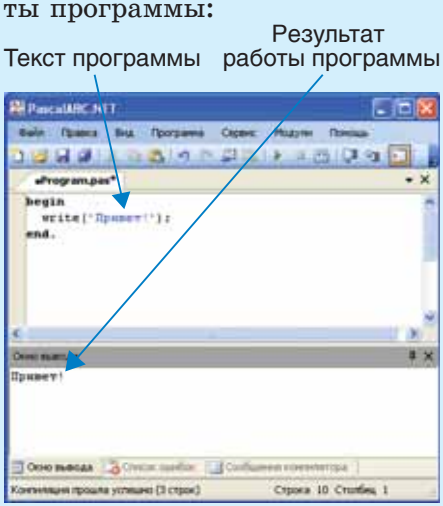
Текст, который нужно вывести на экран, заключают в апострофы (одинарные кавычки). Этот текст не анализируется и выводится в том виде, в котором он записан. Текст можно записать на любом языке. Текстом может быть произвольный набор символов.

В одной программе может быть несколько команд вывода. Для

По традиции, начавшейся в 1978 г. с примера из книги Б. Кернигана и Д. Ритчи «Язык программирования Си», первая программа на любом языке программирования должна выводить на экран приветствие миру:



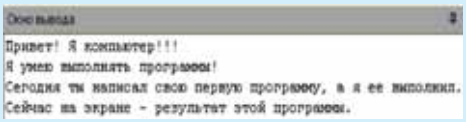
Пример 14.1. Окно среды PascalABC с результатом работы программы:



Пример 14.2. Текст программы:

```
begin
  write('Привет!  ');
  writeln('Я  компьютер!!!');
  write('Я умею выполнять ');
  writeln('программы!');
  write('Сегодня  ты  ');
  write('написал  свою  ');
  write('первую программу,');
  writeln(' а я ее выполнил.');
  write('Сейчас на экране -');
  writeln('  результат  этой
 программы.');
end.
```

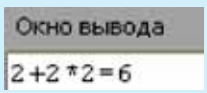
Результат работы программы:



Пример 14.3. Текст программы:

```
begin
  write('2+2*2=');
  write(2+2*2);
end.
```

Результат работы программы:



Две команды `write` в программе можно объединить в одну, отделив текст от выражения запятой:

```
begin
  write('2+2*2=',  2+2*2);
end.
```

вывода текста, записанного в несколько строк, используют команду `writeln( )`. Сочетание «`ln`» (сокр. от англ. *line* — линия, строка), записанное в конце команды, означает, что после вывода нужно перевести курсор в новую строку.

Пример 14.2. Выведем на экран компьютера следующий текст: «`Привет! Я компьютер!!! Я умею выполнять программы! Сегодня ты написал свою первую программу, а я ее выполнил. Сейчас на экране – результат этой программы`».

Используя сочетание команд `write` и `writeln`, текст можно расположить по-разному.

Как вы уже знаете, текст в команде `write( )`, записанный в кавычках, не анализируется. Если кавычки опустить, то производится анализ данных, записанных в скобках. Так, если в скобках написать арифметическое выражение, то сначала вычисляется его значение, а затем выводится результат.

Пример 14.3. Посчитаем значение выражения 2 + 2 * 2.

Если записать выражение в кавычках, то будет выведено само выражение. При отсутствии кавычек на экран будет выведено значение данного выражения.

14.2. Понятие типа данных

На практике редко приходится писать программы, которые решают только одну задачу. Обычно программы пишутся для решения целого класса задач, которые можно сформулировать в общем виде.

С такими задачами вы уже сталкивались в курсе математики. Например, решение задачи «Найдите площадь прямоугольника» можно записать так: $S = a \cdot b$, где переменные a и b обозначают соответственно длину и ширину прямоугольника, а S — площадь. Зная эту формулу, можно найти площадь любого прямоугольника.

В программировании для решения задач в общем виде также используют переменные. Поскольку с такими переменными будет работать компьютер, то они должны храниться в его памяти.

Информацию, представленную в пригодном для обработки на компьютере виде, называют **данными.**

Переменная в программировании — это именованная ячейка памяти, хранящая значение переменной.

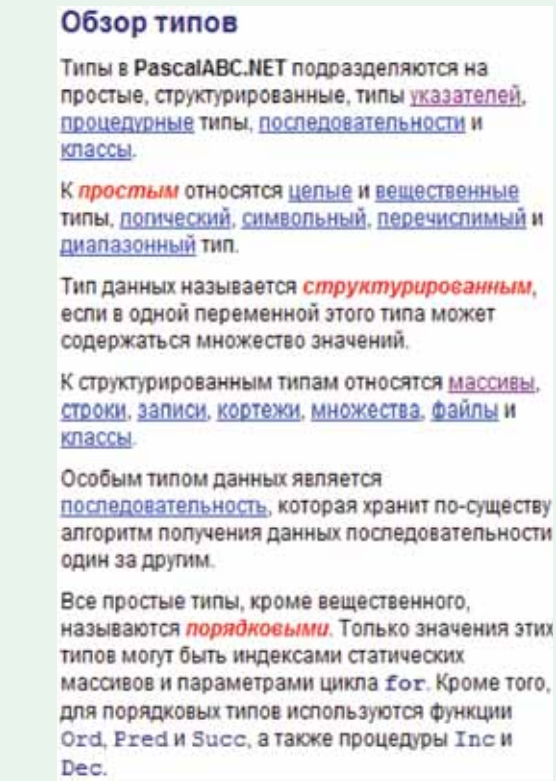
До начала 1950-х гг. XX в. программисты ЭВМ при создании программ пользовались машинным кодом. Запись программы на машинном коде состояла из единиц и нулей. Машинный код принято считать языком программирования первого поколения. Типы данных не использовались.

Первым языком программирования, в котором появилась возможность создавать переменные, считается Ассемблер. В этом языке вместо машинных кодов стали использовать команды, записанные текстом. Ассемблер относится к языкам программирования второго поколения.

В 1957 г. появился язык Фортан, открывший эру языков программирования третьего поколения. Он позволил использовать разные числовые типы данных, необходимые для сложных расчетов: целые, вещественные (действительные) и комплексные.

Дальнейшее развитие языков программирования позволило добавить возможность работы с другими типами данных. Современные языки программирования позволяют работать с большим количеством типов данных.

В среде программирования PascalABC реализовано более 30 различных типов данных.



Обзор типов

Типы в **PascalABC.NET** подразделяются на простые, структурированные, типы указателей, процедурные типы, последовательности и классы.

К *простым* относятся целые и вещественные типы, логический, символьный, перечислимый и диапазонный тип.

Тип данных называется *структурированным*, если в одной переменной этого типа может содержаться множество значений.

К структурированным типам относятся массивы, строки, записи, кортежи, множества, файлы и классы.

Особым типом данных является последовательность, которая хранит по-существу алгоритм получения данных последовательности один за другим.

Все простые типы, кроме вещественного, называются *порядковыми*. Только значения этих типов могут быть индексами статических массивов и параметрами цикла `for`. Кроме того, для порядковых типов используются функции `Ord`, `Pred` и `Succ`, а также процедуры `Inc` и `Dec`.

Пример 14.4. Примеры описания переменных:

```
var x: real;
var x1, y1: real;
var a_1, a_2, a_3: real;
```

Диапазон возможных значений типа `real` задается числами в стандартном представлении от $-1.8 \cdot 10^{308}$ до $1.8 \cdot 10^{308}$. Наименьшее положительное число типа `real` приблизительно равно $5.0 \cdot 10^{-324}$. При вычислениях в числе хранится до 16 цифр.

Компьютер может обрабатывать данные разных типов: целые и действительные числа, символы, тексты и др.

Тип данных определяет способ хранения данных в памяти компьютера, диапазон возможных значений данных и операции, которые с этим типом данных можно выполнять.

Чтобы использовать какую-либо переменную, ее нужно описать. Описание переменных выполняется до начала программы (команды **begin**) (пример 14.4). При описании переменной выделяется память для хранения ее значения. В процессе выполнения программы значение переменной может изменяться.

Для описания переменных используется команда **var** (сокр. от англ. *variable* — переменная).

Формат записи команды:

var <имя переменной>: <тип>;

Для обозначения имени переменной используют буквы латинского алфавита, цифры и знак «_». Первым символом должна быть буква или знак подчеркивания.

Тип данных `real` в языке Pascal позволяет работать с числами и выполнять над ними арифметические действия.

14.3. Оператор присваивания

Одной из основных команд для обработки данных в программе является оператор присваивания.

Оператор присваивания предназначен для того, чтобы:

- задавать значения переменным;
- вычислять значения арифметического выражения (результат вычисления будет записан как значение переменной).

Формат записи оператора:

<имя переменной>:= <выражение>;

(Рассмотрите пример 14.5.)

В записи арифметического выражения используются знаки математических действий.

Математические операции	Запись в Pascal
+ (сложение)	+
− (вычитание)	-
· (умножение)	*
: (деление)	/

Приоритет выполнения операций соответствует принятому в математике: сначала выполняются умножение и деление, а затем сложение и вычитание. Для изменения порядка действий в выражениях используют скобки.

Пример 14.5. Примеры записи оператора присваивания:

```
x:= 7;
x1:= 3.5;
a_1: = 20 * (x + x1) - 32;
y:= y + 7;
```

Пример 14.6. Запишем оператор присваивания на Pascal для математических выражений:

Выражение	Запись на Pascal
$S = 2(a + b)$	`S:=2*(a+b);`
$S = a^2$	`S:=a*a;`
$a = \frac{x + y}{3}$	`a:=(x+y)/3;`

Пример 14.7. Запишем оператор присваивания, после выполнения которого значение переменной a увеличится в 2 раза, а переменной b уменьшится на 3.

В Pascal допустимы команды присваивания следующего вида:

```
a:= a * 2;
```

Смысл такой команды следующий: из ячейки памяти извлекается значение переменной a, затем оно умножается на 2, результат записывается в ту же ячейку памяти. Старое значение переменной a будет потеряно.

Запись оператора присваивания для изменения значения переменной b следующая:

```
b:= b - 3;
```

В PascalABC.NET определены операторы присваивания со значками +=, −=, *=, /=. Они позволяют изменить значение переменной. Например:

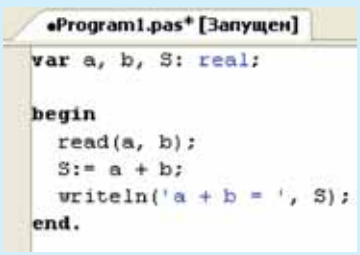
```
a *= 2;  // увеличить a в 2 раза;
b -= 3;  // уменьшить b на 3.
```

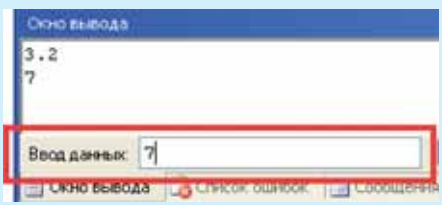
Пример 14.8. Ввести два числа, найти и вывести их сумму.

Текст программы:

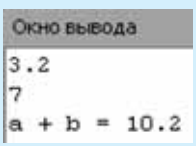
```
var a, b, S: real;
begin
 read(a, b);
 S:= a + b;
 writeln('a + b =', S);
end.
```

Ввод данных:





Результат:



Для записи обыкновенной дроби используется знак деления. Знак умножения опускать нельзя. Целая часть дробного числа отделяется от дробной части точкой. (Рассмотрите примеры 14.6 и 14.7 на с. 93.)

14.4. Ввод данных

Начальные значения переменным можно задавать не только с помощью оператора присваивания, но и путем ввода с клавиатуры. В этом случае, если необходимы вычисления с новым набором значений исходных данных, текст программы не нужно изменять.

Команда `read( )` предназначена для **ввода данных**. В скобках через запятую перечисляются имена переменных, значения которых необходимо ввести.

Ввод данных происходит в нижней части окна программы PascalABC. Для этого используется окно «Ввод данных». После нажатия кнопки «Ввести» или клавиши «Enter» введенные значения переносятся в окно вывода. После завершения работы программы в этом же окне будет выведен результат (пример 14.8).

14.5. Структура программы

Все программы на языке программирования Pascal имеют общую структуру. В программе можно выделить следующие разделы:

- заголовок (не обязателен);
- подключаемые библиотеки (модули) (если подключать дополнительные библиотеки не нужно, раздел отсутствует; известные библиотеки: `Drawman`, `Robot`, `RobTasks`);
- описание переменных с указанием их типа;
- описание вспомогательных алгоритмов (если использовать вспомогательные алгоритмы не нужно, раздел отсутствует);
- **begin** ... **end.** — служебные слова, обрамляющие тело основной программы, в которой находятся исполняемые команды; **begin** начинает исполняемую часть программы, а **end.** (точка в конце обязательна) ее завершает.

В минимально возможном наборе программа состоит из пустого тела программы: **begin end.** Программа, содержащая все разделы, представлена в примере 14.9.

Для каждого раздела определено ключевое служебное слово, которым он начинается. При написании программы ключевые слова выделяются полужирным шрифтом.

Пример 14.9. Программа, содержащая все разделы (подсчитывается количество закрашенных клеток в поле Робота размером 10 × 10):

```
//заголовок программы
//(необязательно)
program Primer;
//описание библиотек
uses Robot, RobTasks;
//описание переменных
var k: integer;
//описание вспомогательных
//алгоритмов
procedure line;
begin
 for var i:= 1 to 9 do
 begin
  if CellisPainted then
   k:= k + 1;
  right;
 end;
 if CellisPainted then
  k:= k + 1;
end;
procedure back;
begin
 for var i:= 1 to 9 do
  left;
end;
//основная программа
begin
//тело программы
 Task('myrob11');
 k:= 0;
 for var i:= 1 to 9 do
 begin
  line; back; down;
 end;
 line;
 writeln(k);
end.
```

1. Какая команда языка программирования Pascal предназначена для вывода данных?
2. Что определяет тип данных?
3. Для чего используется команда присваивания?
4. Какая команда языка программирования Pascal предназначена для ввода данных?
5. Из каких разделов состоит программа на языке программирования Pascal?

Упражнения

1 Для программы из примера 14.2 выполните следующие задания (файл с программой можно скачать):

1. Замените все команды `writeln` на команды `write` и выполните программу. Что произошло? Объясните почему.
2. Как изменится результат работы программы, если в исходном тексте заменить все команды `write` на `writeln`?
3. Измените программу так, чтобы текст на экране выглядел следующим образом:

```
Привет! Я компьютер!!! Я умею выполнять программы!
Ты сегодня написал свою первую программу!!!
Я выполнил твою программу. Посмотри на экране результат!
```

2 Внесите необходимые изменения в программу из примера 14.3, чтобы действия выполнялись в том порядке, в котором записаны, т. е. сначала сложение, а потом умножение.

3 Вводится возраст пользователя в годах. Определите возраст пользователя через 5 лет.

4 Напишите программу, в которой вводятся два числа a и b. Затем первое число уменьшается в 2 раза, а второе увеличивается на 30. Выведите измененные значения переменных.

5 Напишите программу для вычисления значения числовых выражений:

1. $23 + 45 \cdot 11 - 15$.
2. $\dfrac{37 + 2 \cdot 27}{41}$.
3. $\dfrac{5638 - 2347}{49} + \dfrac{123 \cdot 756}{4455}$.

§ 15. Организация вычислений

При решении любой задачи человеку приходится выполнять следующие действия:

- определение исходных данных (что дано в задаче);
- определение результатов (что нужно получить);
- обработка исходных данных в соответствии с известными правилами так, чтобы получить результат.

Применяя указанные правила к решению задачи по программированию, получим следующие этапы решения задачи:

I. Определение исходных данных.

II. Определение результатов.

II. Составление алгоритма решения задачи.

IV. Определение типов данных для переменных, используемых при реализации алгоритма.

V. Написание программы.

VI. Тестирование программы.

VII. Анализ результатов.

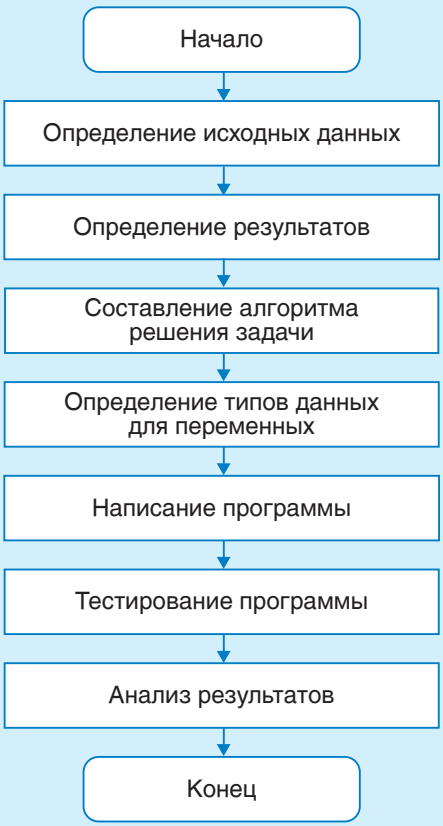
(Рассмотрите пример 15.1.)

Тестирование программы — проверка правильности работы программы при разных наборах исходных данных.

Пример 15.1. Решение задач по физике принято оформлять определенным образом.

Слева записывается то, что дано и что нужно получить, справа — последовательность действий, приводящая к решению задачи. Аналогично оформляются решения задач по химии, геометрии.

Этапы решения задачи по программированию можно представить следующим образом:



Пример 15.2.

V. Программа:

```
var  x, y, z, a:  real;
begin
 write('введите  x  =  ');
 read(x);
 write('введите  y  =  ');
 read(y);
 write('введите  z  =  ');
 read(z);
 a:=(2*x+3*y-z)/(3+x*x);
 writeln('a  =  ',a);
end.
```

VI. Тестирование программы.

Запустите программу и введите значения: $x = 2$, $y = 3$, $z = 1$.

Результат работы программы должен быть следующим:

```
Окно вывода
введите x = 2
введите y = 3
введите z = 1
a = 1.71428571428571
```

А для значений $x = 2$, $y = 4$, $z = 2$ получим:

```
Окно вывода
введите x = 2
введите y = 4
введите z = 2
a = 2
```

VII. Проверка правильности вычислений может быть выполнена на калькуляторе.

15.1. Вычисление значения арифметического выражения

Пример 15.2. Даны переменные x, y, z. Напишем программу для вычисления значения выражения $a = \dfrac{2x + 3y - z}{3 + x^2}$.

Этапы выполнения задания:

I. Определение исходных данных: переменные x, y, z.

II. Определение результатов: переменная a.

III. Алгоритм решения задачи:

1. Ввод исходных данных.
2. Вычисление значения выражения.
3. Вывод результата.

IV. Описание переменных.

Все переменные, определенные для решения задачи, имеют тип `real`.

В приведенном примере перед каждой командой ввода записана команда вывода с пояснениями о том, значение какой переменной нужно вводить.

При написании программ для вычисления значения арифметического выражения часто допускают следующие ошибки:

`2x+3;` `(x-4)(x+2);`	Пропущен знак *
`(2+y)/(x*x;`	Пропущена скобка

Будьте внимательны!

15.2. Использование языка программирования для решения задач

Пример 15.3. Напишем программу для решения геометрической задачи. Задан квадрат с длиной стороны a. Требуется найти его площадь и периметр.

Этапы выполнения задания:

I. Определение исходных данных: переменная a (длина стороны).

II. Определение результатов: переменные S (площадь) и P (периметр).

III. Алгоритм решения задачи:

1. Ввод исходных данных.

2. Вычисление значений площади в математике производится по формуле $S = a^2$, а периметра — по формуле $P = 4a$. В программе этим формулам будут соответствовать команды присваивания: S:= a * a; P:= 4 * a.

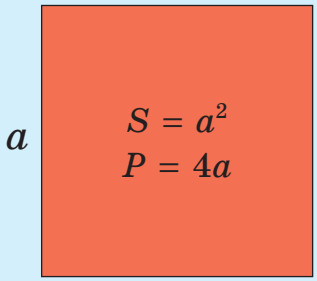
3. Вывод результата.

IV. Описание переменных:

Все переменные, определенные для решения задачи, имеют тип `real`.

Обратите внимание: запись формул в операторе присваивания может отличаться от записи математических формул.

Пример 15.3.



V. Программа:

```
var a,S,P: real;
begin
 write('введите a = ');
 read(a);
 s:= a*a;
 p:= 4*a;
 writeln('площадь = ',S);
 writeln('периметр = ',P);
end.
```

VI. Тестирование программы.

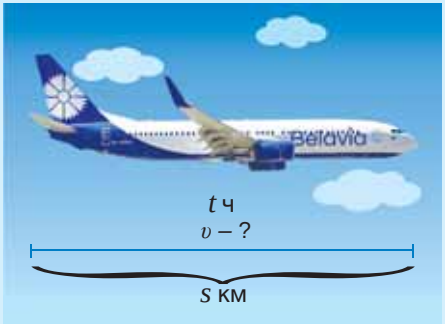
Запустите программу и введите значение a = 5.2.

Результат работы программы должен быть следующим:

```
Окно вывода
введите a = 5.2
площадь= 27.04
периметр= 20.8
```

VII. Проверка правильности вычислений может быть выполнена на калькуляторе.

Пример 15.4.



V. Программа:

```
var s, t, v: real;
begin
 write('введите s = ');
 read(s);
 write('введите t = ');
 read(t);
 v:= s / t;
 writeln('скорость = ', v);
end.
```

VI. Тестирование программы.

Запустите программу и введите значения $s = 3550$ и $t = 4$.

Результат работы программы должен быть следующим:

Окно вывода

```
введите s = 3550
введите t = 4
скорость = 887.5
```

VII. Проверка правильности вычислений может быть выполнена на калькуляторе.

Пример 15.4. Напишем программу для решения физической задачи. Расстояние между двумя городами составляет s км. Самолет пролетает это расстояние за t ч. Определите скорость самолета.

Этапы выполнения задания:

I. Определение исходных данных: переменные s (расстояние) и t (время).

II. Определение результатов: переменная v (скорость).

III. Алгоритм решения задачи:

1. Ввод исходных данных.

2. Согласно формуле расстояния: $s = vt$. Отсюда выразим v: $v = \frac{s}{t}$.

3. Вывод результата.

IV. Описание переменных:

Все переменные, определенные для решения задачи, имеют тип `real`.

При написании программ обращайте внимание на форматирование их текста:

- в первой позиции на экране пишут только слова **var**, **begin** **end**, а остальные со сдвигом на 2—4 позиции вправо;
- если в программе несколько частей, то их можно отделить друг от друга пустой строкой.

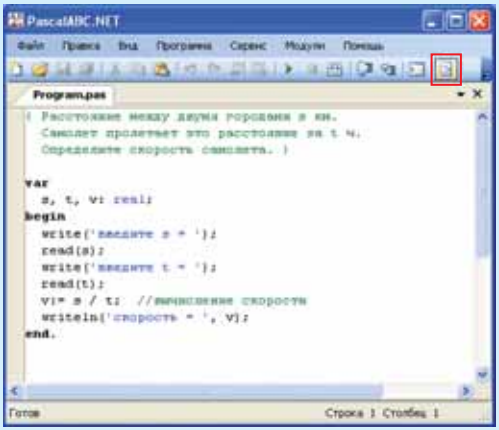
Выполнение этих правил повышает читаемость программы.

В программе можно использовать **комментарии** — текст, который не анализируется при запуске программы на выполнение.

Текст после символов // считается комментарием и выделяется зеленым цветом (пример 15.5).

В комментариях удобно записывать условие задачи. Пояснения к командам, записанные как комментарии, нужны для понимания действия, выполняемого командой. В языке программирования Pascal комментарии можно записать в несколько строк. Тогда текст, являющийся комментарием, заключают в фигурные скобки.

Пример 15.5. Программа с комментариями и отформатированным кодом:



Значок для форматирования кода на панели инструментов имеет следующий вид:

1. Перечислите этапы решения задачи по программированию.
2. Что понимают под тестированием программы?
3. Для чего можно использовать комментарии?

Упражнения

1 Даны x, y, z. Напишите программу для вычисления значения арифметических выражений.

1. $a = \dfrac{x + y - z}{x^2 + 2}$. 2. $a = 5\dfrac{2x - z}{3 + y^2}$. 3. $a = (1 + z)\dfrac{x + \dfrac{y}{x^2 + 4}}{2 + \dfrac{1}{x^2 + 4}}$.

2 Напишите программу для решения геометрической задачи.

1. Найдите длину окружности и площадь круга заданного радиуса.

2*. Найдите угол при основании равнобедренного треугольника, если известен угол при вершине.

3 Напишите программу для решения физической задачи.

1. Велосипедист едет с постоянной скоростью v км/ч. За сколько минут он проедет расстояние в s км?

2*. Автомобиль проходит первую часть пути длиной s_1 км за t_1 мин, участок пути длиной s_2 км за t_2 мин и участок длиной s_3 км за t_3 мин. Найдите среднюю скорость автомобиля в км/ч.

4 Напишите программу для решения химической задачи.

1. В организме человека на долю атомов кислорода приходится 65 % от массы тела. Найдите массу атомов кислорода для своей массы тела.

2*. Масса атома кислорода равна $26.56 \cdot 10^{-27}$ (это число на языке Pascal записывается так: 26.56E-27, буква E — английская). Сколько атомов кислорода содержится в вашем теле?

§ 16. Реализация алгоритмов работы с целочисленными данными

В PascalABC определены различные типы данных для работы с целыми числами, позволяющие выполнять действия над данными из разных числовых диапазонов. Чем больше диапазон, тем больше места в памяти компьютера отводится для хранения переменных.

Некоторые целочисленные типы данных:

Тип	Диапазон значений
`shortint`	–128..127
`smallint`	–32768..32767
`integer`, `longint`	–2147483648..2147483647
`byte`	0..255
`word`	0..65535

16.1. Целочисленный тип данных

Часто при решении задач нужно работать с целыми числами. Для этого в Pascal используется тип данных `integer`. С помощью переменных этого типа можно задавать целые числа из диапазона от –2147483648 до 2147483647. Для типа данных `integer` определены следующие операции:

Математические операции	Запись в Pascal
+ (сложение)	+
– (вычитание)	–
· (умножение)	*
целочисленное деление	`div`
нахождение остатка	`mod`