

## Глава 3

# ОСНОВЫ АЛГОРИТМИЗАЦИИ И ПРОГРАММИРОВАНИЯ

### § 13. Основные алгоритмические конструкции

#### 13.1. Алгоритм и алгоритмические конструкции

В 7-м классе вы познакомились с основными алгоритмическими конструкциями. Для решения задач по программированию были выделены основные этапы (пример 13.1).

**Алгоритм** — конечная последовательность точных действий, формальное выполнение которых позволяет получить решение задачи для любого допустимого набора исходных данных.

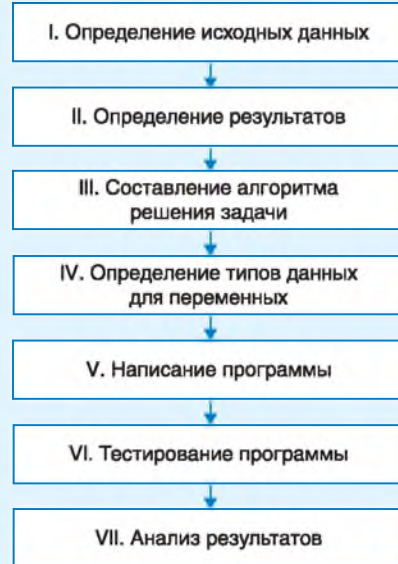
**Исполнитель** — человек, группа людей или техническое устройство, которые способны правильно выполнять команды алгоритмов. В дальнейшем будем рассматривать только исполнителя-устройство с ограниченным набором команд. Набор команд одного исполнителя называют **системой команд исполнителя**. Команды компьютерного исполнителя могут быть реализованы в виде процедур и функций.

Все команды исполнителя делят на группы:

1. Команды, которые непосредственно выполняет исполнитель.
2. Команды, изменяющие порядок выполнения других команд исполнителя.

Любой алгоритм может быть записан с использованием трех базовых алгоритмических конструкций:

**Пример 13.1.** Этапы решения задачи по программированию:



В процессе решения задачи некоторые этапы приходится повторять до тех пор, пока анализ результатов не покажет, что задача решена верно.

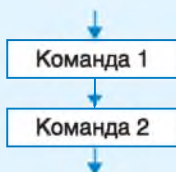
Для поиска ошибок можно использовать средства отладки программ (см. Приложение 3, с. 161—162).

В 1966 г. итальянские математики Коррадо Бём (1923—2017) и Джузеппе Джакопини (1936—2001) сформулировали и доказали положение структурного программирования, согласно которому любой исполняемый алгоритм может быть преобразован к структурированному виду, т. е. виду, когда ход выполнения алгоритма определяется при помощи трех структур управления: последовательной, ветвлений и циклов.

Полностью концепция структурного программирования была разработана в середине 70-х гг. при участии Э. Дейкстры.

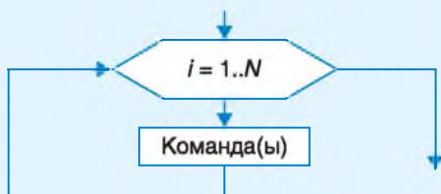
**Пример 13.2.** Блок-схемы алгоритмических конструкций:

1. Следование:

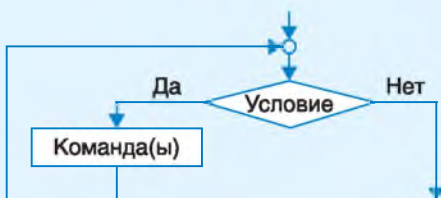


2. Цикл:

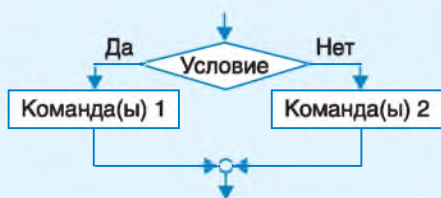
1) цикл с параметром (значение параметра изменяется от 1 до  $N$ )



2) цикл с предусловием



3. Команда ветвления



**следование, цикл и ветвление** (пример 13.2).

Команды цикла и ветвления управляют порядком выполнения других команд в программе и относятся к **командам управления (управляющим конструкциям)**.

Последовательность команд, исполнителем которой является компьютер, называется **программой**. Программа представляет собой запись на некотором формальном языке — языке программирования. Командами в языке программирования считают:

- операторы (оператор присваивания, оператор ветвления, оператор цикла и др.);
- вызовы вспомогательных алгоритмов (как встроенных в библиотеки, так и созданных пользователем).

### 13.2. Алгоритмическая конструкция *следование*

Алгоритмическая конструкция *следование* — последовательность команд алгоритма, которые выполняются в том порядке, в котором они записаны. Среди команд, образующих алгоритмическую конструкцию *следование*, отсутствуют команды, меняющие порядок выполнения других команд.

В 7-м классе, изучая язык Pascal, вы использовали следующие команды (пример 13.3):

- процедуры для ввода и вывода данных;
- оператор присваивания.

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

Для **вывода данных** используют команду `write()`. Она позволяет вывести текстовые сообщения и числовые значения. Текстовые сообщения записываются в кавычках, выводятся в виде последовательности символов так, как записаны, и не анализируются при выполнении.

При использовании команды `writeln()`; после вывода сообщения или числа происходит перевод курсора на следующую строку.

**Оператор присваивания** предназначен для того, чтобы:

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

Формат записи оператора присваивания:

`<имя переменной> := <выражение>;`

В записи арифметического выражения используются знаки математических действий: сложения (+), вычитания (−), умножения (\*), деления (/), а также целочисленного деления (div) и нахождения остатка (mod). Следует помнить, что операция деления (/) используется при вычислениях с данными типа `real`. Для данных типа `integer` используются операции `div` и `mod`.

Нередко в одной программе приходится выполнять одну и ту же последовательность команд несколько раз. В этом случае удобно использовать

**Пример 13.3.** Даны  $x, y$ . Написать программу для вычисления значения выражения

$$a = \frac{2x}{7 + y^2}(x - y).$$

Этапы выполнения задания

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

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`.

V. Программа:

```
var x,y,a: real;
begin
  write('Введите x = ');
  read(x);
  write('Введите y = ');
  read(y);
  a:= 2 * x * (x-y)/(7 + y * y);
  writeln('a =',a);
end.
```

End.

VI. Тестирование программы:

Запустить программу и ввести значения  $x = 3.8, y = 2.7$ . Результат:

#### Окно вывода

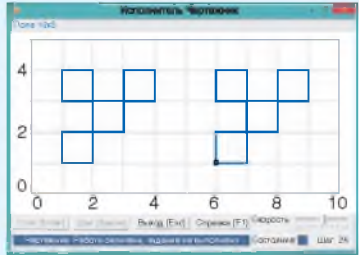
```
Введите x = 3.8
Введите y = 2.7
a = 0.585024492652204
```

VII. Правильность вычислений проверить на калькуляторе.

В примере алгоритмическая конструкция *следование* образована командами:

- вывод сообщений;
- ввод значений переменных;
- команда присваивания;
- вывод результата.

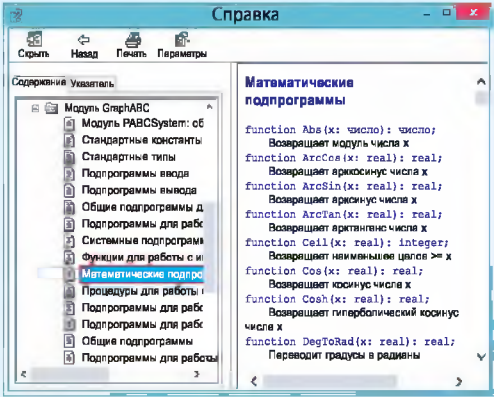
**Пример 13.4.** Написать программу для вывода изображения:



Изображение состоит из двух одинаковых фигур. Оформим вспомогательный алгоритм `Figura` для изображения одной фигуры. Программа:

```
uses Drawman;
procedure Figura;
begin
  PenDown; OnVector(1, 0);
  OnVector(0, 3); OnVector(-1, 0);
  OnVector(0, -1); OnVector(3, 0);
  OnVector(0, 1); OnVector(-1, 0);
  OnVector(0, -2); OnVector(-2, 0);
  OnVector(0, -1); PenUp;
end;
begin
  Field(10, 5);
  ToPoint(1, 1); Figura;
  ToPoint(6, 1); Figura;
end.
```

**Пример 13.5.** Перечень математических функций можно посмотреть в справке PascalABC:



вспомогательный алгоритм, который можно выполнять нужное число раз, обращаясь к его названию.

**Вспомогательный алгоритм** — алгоритм, который можно использовать в других алгоритмах, указав его имя и, если необходимо, значения параметров.

Вспомогательный алгоритм решает некоторую часть основной задачи. Вызов вспомогательного алгоритма является командой, которая может заменять несколько команд.

Вспомогательные алгоритмы вы использовали при написании программ для учебных компьютерных исполнителей Чертежник и Робот (пример 13.4). Команды `read` и `write` тоже реализованы как вспомогательные алгоритмы.

При вычислениях часто используются различные математические функции (пример 13.5). Эти функции реализованы как встроенные вспомогательные алгоритмы и могут применяться при записи арифметических выражений. Аргументы функций всегда записываются в скобках. Некоторые из функций приведены в таблице (другие можно посмотреть в *Приложении 3*, с. 158).

Запись на языке Pascal	Описание
<code>abs (x)</code>	Находит модуль числа $x$
<code>sqr (x)</code>	Возводит число $x$ в квадрат
<code>sqrt (x)</code>	Находит корень квадратный из числа $x$ . Результат — всегда число типа <code>real</code>



Запись на языке Pascal	Описание
<code>trunc(x)</code>	Находит целую часть действительного числа $x$ (real). Результат — число типа integer
<code>frac(x)</code>	Находит дробную часть действительного числа $x$ (real). Результат — число типа real
<code>sin(x)</code>	Вычисляет синус числа $x$ . Число $x$ задается в радианах <sup>1</sup>
<code>cos(x)</code>	Вычисляет косинус числа $x$ . Число $x$ задается в радианах
<code>RadToDeg(x)</code>	Переводит радианы в градусы
<code>DegToRad(x)</code>	Переводит градусы в радианы

Аргументом функции может быть число, переменная, выражение или другая функция: `sin(DegToRad(45))`, `sqrt(abs(-16))`.

В примере 13.6 используются математические функции для возведения числа в квадрат и вычисления квадратного корня.

**Пример 13.6.** Задана длина стороны квадрата  $a$ . Написать программу нахождения площади квадрата и длины его диагонали.

Этапы выполнения задания

I. Исходные данные: длина стороны, переменная  $a$ .

II. Результат: переменные  $S$  (площадь) и  $d$  (длина диагонали).

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Вычисление площади по формуле  $S = a^2$ .

3. Вычисление длины диагонали по формуле  $d = a\sqrt{2}$ .

4. Вывод результата.

IV. Описание переменных: все переменные, определенные для решения задачи, имеют тип real.

V. Программа:

```
var a, S, d: real;
begin
  write('Введите a ='); read(a);
  S:= sqr(a); d:= a*sqrt(2);
  writeln('S=',s); writeln('d=',d);
end.
```

VI. Тестирование программы.

Запустить программу и ввести значение  $a = 5.6$ . Результат:

#### Окно вывода

```
Введите a = 5.6
S = 31.36
d = 7.91959594928933
```

Правильность вычислений можно проверить на калькуляторе.



1. Что такое алгоритм?
2. Перечислите основные алгоритмические конструкции.
3. Какая команда используется в языке Pascal для ввода данных?
4. Какие команды используются в языке Pascal для вывода данных?
5. Для чего нужна команда присваивания?
6. В каких случаях удобно использовать вспомогательный алгоритм?
7. Какие математические функции могут использоваться при записи арифметических выражений?

<sup>1</sup> Радиан — единица измерения углов в Международной системе единиц (1 радиан соответствует величине развернутого угла  $180^\circ$ ).

**Упражнения**

1 Расставьте команды программы в правильном порядке так, чтобы можно было вычислить значение выражения  $a = \frac{2x}{x^2 + 4}$ .

1. `writeln('a = ', a);`

2. `write('Введите значение x = ');`

3. `End.`

4. `Var x, y, a: real;`

5. `Begin`

6. `a := 2 * x / (x * x + 4);`

7. `read(x);`

2 Определите типы данных для каждой переменной, использованной в операторе присваивания.

1. `y := sqrt(a - 4) / 16;`

2. `z := sqr(3 * a + 2);`

3. `a := abs(a - 4.2);`

4. `d := x mod 2;`

5. `y := int(a);`

6. `y := trunc(a);`

7. `y := frac(a);`

8. `s := sin(3.14 * r).`

3 Найдите и исправьте ошибки в программах.

1. `var x, y, z1, z2: integer;`

`begin`

`write('Введите x =');`

`read(x);`

`write('Введите y =');`

`read(y);`

`z1 := int(x/y);`

`z2 := frac(x/y);`

`write('Целая часть =', z1);`

`write('Дробная часть =', z2);`

`end.`

2. `var x, y, z1, z2: real;`

`begin`

`write('Введите x =');`

`read(x);`

`write('Введите y =');`

`read(y);`

`z1 := x div y;`

`z2 := x mod y;`

`write('Целая часть =', z1);`

`write('Остаток =', z2);`

`end.`

4 Даны  $x$  и  $z$ . Измените программу из примера 13.4 так, чтобы вычислялось значение выражения  $a = \frac{2x}{\sqrt{z^2 + 9}}$ .

5 Заданы три числа. Напишите программу для нахождения среднего арифметического этих чисел.

6 Заданы два числа. Напишите программу для нахождения частного от деления первого числа на второе и округлите результат до ближайшего целого.

7 Даны гипотенуза и катет прямоугольного треугольника. Напишите программу для нахождения второго катета и площади треугольника.

8\* Заданы два числа. Напишите программу для нахождения частного этих чисел. Округлите результат до десятых, оставив в дробной части одну цифру.

## § 14. Графические возможности среды программирования PascalABC

### 14.1. Основы работы с графикой

Вы уже знакомы с графическими редакторами, в которых для построения изображений на компьютере используются графические примитивы — простые геометрические фигуры: прямоугольник, окружность, эллипс, отрезок и т. д. Графический редактор — программа, написанная на каком-либо языке программирования.

Для работы с графикой языки программирования используют специальные библиотеки (модули), содержащие наборы команд для построения изображений. В PascalABC для работы с графикой используется библиотека GraphABC. Для подключения этой библиотеки в программе записывается команда `uses GraphABC;`

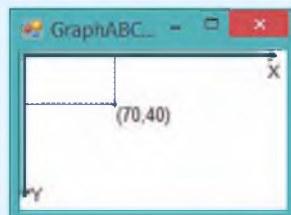
Положение фигур задается координатами в графическом окне. Координатная плоскость в нем отличается от той, которую вы используете на уроках математики. Началом координат является верхний левый угол графического окна — точка  $(0; 0)$  (пример 14.1). Координаты задают порядковый номер пикселя по горизонтали и вертикали, поэтому они могут быть только целыми числами. Отсчет значений координаты  $x$  происходит слева направо, а координаты  $y$  — сверху вниз. По умолчанию создается графическое окно размером  $640 \times 480$  пикселей.



Первые компьютеры не имели возможностей работы с графикой. На печатающих устройствах выводились «картинки», состоящие из символов<sup>1</sup>.

В 1958 г. был запущен компьютер Lincoln TX-2, впервые использующий графический экран. В 1981 г. начали применять цвета. В графическом режиме при разрешении  $320 \times 200$  пикселей использовались 4 цвета из стандартных палитр: пурпурный, сине-зеленый, белый, черный или красный, зеленый, желтый, черный.

**Пример 14.1.** Графическое окно среды программирования PascalABC с изображением координатных осей и точки с координатами  $(70, 40)$ .

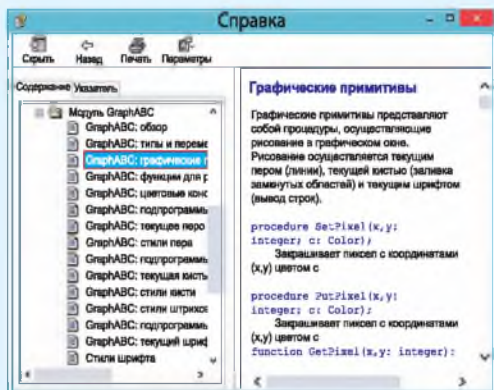


Точка расположена на расстоянии 70 пикселей от левого края окна и на расстоянии 40 пикселей от верхнего края.

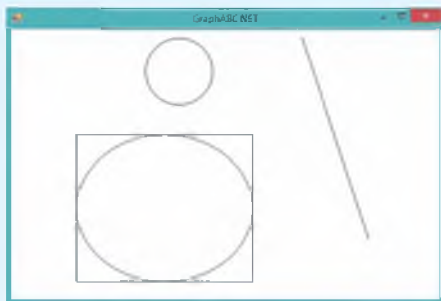
Размеры графического окна можно задать командой `SetWindowSize(n,m);` В скобках указаны размеры окна по горизонтали и вертикали.

<sup>1</sup> <https://www.asciart.eu/buildings-and-places/houses> (дата доступа: 26.07.2018).

**Пример 14.2.** Работа со справочной системой. Для перехода в справочник необходимо нажать клавишу F1 или выполнить команду меню: **Помощь** → **Справка**. В открывшемся окне перейти в раздел **Стандартные модули** и выбрать **Модуль GraphABC**.



**Пример 14.3.** Графические примитивы:



Программа для их рисования:

```
uses GraphABC;
begin
    //Круг
    Circle(250, 125, 30);
    //Прямоугольник
    Rectangle(100,200,400,450);
    //Эллипс
    Ellipse(100,200,400,450);
    //Отрезок
    Line(450, 50, 550, 350);
end.
```

## 14.2. Работа со справочной системой среды программирования PascalABC

В библиотеке GraphABC содержится большое количество команд. Эти команды описаны в справочной системе среды PascalABC (пример 14.2). Здесь есть описание графических примитивов, названия цветовых констант, описание работы с пером и кистью, команды работы с графическим окном.

Команды библиотеки GraphABC — вспомогательные алгоритмы, записанные как отдельные процедуры. Использование команды в программе означает вызов соответствующего алгоритма.

## 14.3. Основные графические примитивы

Рассмотрим команды для рисования графических примитивов:

- **Line(x1,y1,x2,y2)** — отрезок, соединяющий точки с координатами (x1, y1) и (x2, y2).
- **MoveTo(x,y)** — устанавливает текущую позицию рисования в точку (x, y);
- **LineTo(x,y)** — отрезок от текущей позиции до точки (x, y);
- **Rectangle(x1,y1,x2,y2)** — прямоугольник, заданный координатами противоположных вершин (x1, y1) и (x2, y2);
- **Circle(x1,y1,r)** — круг с центром в точке (x1, y1) и радиусом r;
- **Ellipse(x1,y1,x2,y2)** — овал (эллипс), вписанный в прямоугольник с координатами противоположных вершин (x1, y1) и (x2, y2).

(Рассмотрите пример 14.3.)

Команды для рисования других графических примитивов имеются в справочной системе и в *Приложении 3* (с. 159).

**Пример 14.4.** Написать программу, которая строит изображение домика,



используя процедуры `Line`, `LineTo`, `Rectangle`, `Circle`.

Этапы выполнения задания

I. Исходные данные: результат работы программы не зависит от исходных данных.

II. Результат: готовый рисунок.

III. Алгоритм решения задачи.

Рисунок состоит из: прямоугольников, отрезков, круга. Для расчета координат рекомендуется предварительно сделать рисунок на листе бумаги в клеточку.

IV. Описание переменных. Переменные не используются.

#### 14.4. Работа с пером и кистью

В графических редакторах, прежде чем рисовать какие-либо фигуры, устанавливают их цвет. Обычно выбирают два цвета. Цвет 1 определяет цвет линий и контуров фигур, Цвет 2 используется для заливки фигур. Кроме того, можно изменять стиль линий и заливки, а также определять толщину линий.

В графическом режиме PascalABC настройки линии определяет перо (Pen), а настройки внутренней области фигур — кисть (Brush). Команды для работы с кистью и пером приведены в таблице.

Команда	Описание
<code>SetPenColor</code>	Цвет линий
<code>SetPenWidth</code>	Толщина линии
<code>SetPenStyle</code>	Стиль линий
<code>SetBrushColor</code>	Цвет заливки
<code>SetBrushStyle</code>	Стиль заливки
<code>SetBrushHatch</code>	Вид штриховки для заливки

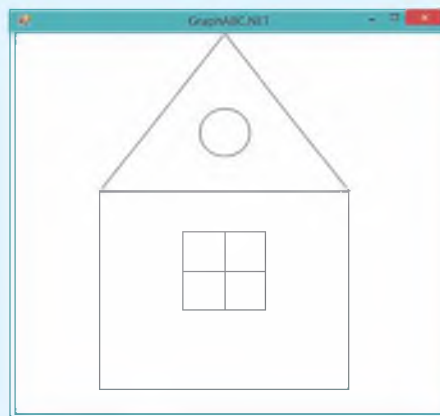
#### Пример 14.4.

V. Программа:

```
uses GraphABC;
begin
    //Дом
    Rectangle(100,200,400,450);
    //Окно
    Rectangle(200,250,300,350);
    Line(250, 250, 250, 350);
    Line(200, 300, 300, 300);
    //Крыша
    MoveTo(100,200);
    LineTo(250, 0);
    LineTo(400, 200);
    Circle(250, 125, 30);
end.
```

VI. Тестирование программы:

Запустить программу. Результат:



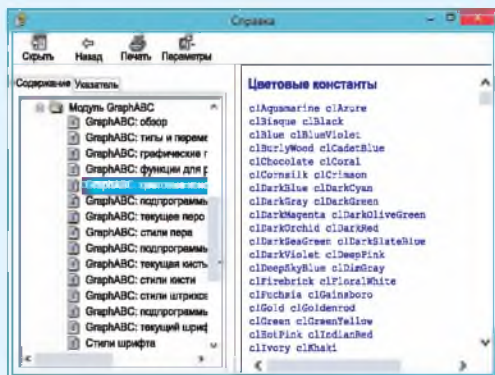
Обратите внимание, что при написании команд у вас появляются подсказки:

```
Circle(
    procedure GraphABC.Circle(x: integer; y: integer; r: integer);
    Описание:
    Рисует заполненную окружность с центром (x,y) и радиусом r
```

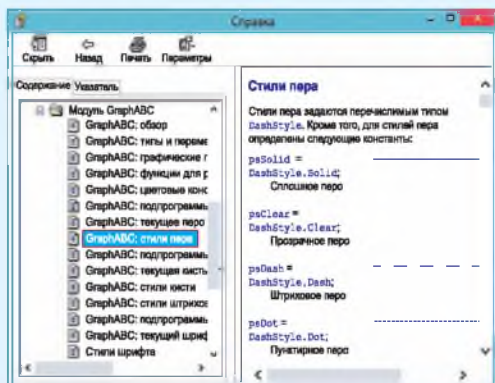
Подсказка появляется также при наведении указателя мыши на уже написанную команду.

Если установить текстовый курсор на команду и нажать F1, то откроется страница из справочника с описанием этой команды.

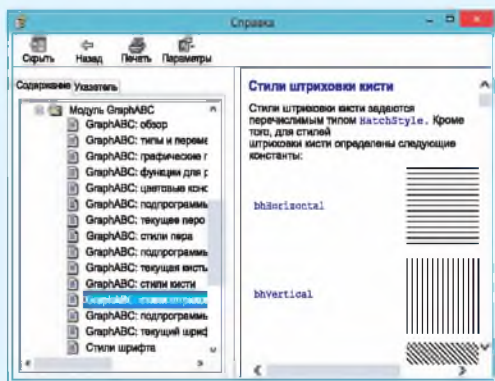
Пример 14.5. Цветовые константы.



Пример 14.6. Стили пера.



Пример 14.7. Стили штриховки кисти.



Значение, которое нужно установить для каждой из команд, записывается в скобках. Например:

- `SetPenColor(clRed)` — красный цвет рисования линий;
- `SetBrushColor(clBlue)` — синий цвет заливки фигур;
- `SetPenWidth(3)` — толщина пера в 3 пикселя;
- `SetPenStyle(psDot)` — штриховая линия;
- `SetBrushStyle(bsHatch)` — штриховая заливка;
- `SetBrushHatch(bhCross)` — штриховка в клеточку.

Значения цветовых констант, стилей линий и заливок можно найти в справочной системе среды PascalABC (примеры 14.5—14.7) и в *Приложении 3* (с. 160).

Команды для установки цвета и стиля записывают перед командой рисования фигуры. Эти команды действуют до тех пор, пока цвет или стиль не будет изменен. Если, например, для пера была установлена толщина в 3 пикселя, то отрезки и границы фигур будут иметь толщину в 3 пикселя до новой смены толщины пера.

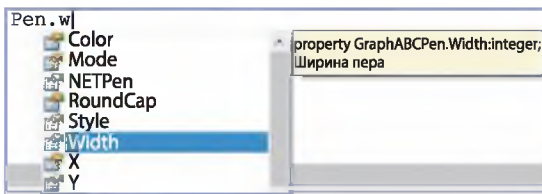
Для фигур по умолчанию установлена заливка белым цветом. Если цвет кисти выбрать до рисования фигуры, то фигура будет закрашена установленным цветом. Цвет уже нарисованной фигуры можно изменить с помощью команды заливки:

`FloodFill(x,y,c);` — заливает ограниченную область одного цвета цветом `c`, начиная с точки внутри области `(x,y)` (в примере 14.8 показано, как раскрашен домик из примера 14.4).

Среда программирования PascalABC позволяет обращаться к свойствам кисти и пера по-другому. Так, например, для изменения цвета (стиля, толщины) можно записать

```
Pen.Color := clRed;
Pen.Style := psDot;
Pen.Width := 3;
```

Подсказка среды выглядит следующим образом:



Вторую часть команды можно выбрать из выпадающего списка.

При изучении векторной графики вы познакомились с цветовой моделью RGB, которая позволяет записать любой цвет тремя составляющими: красной, зеленой и синей. Функция RGB (r, g, b) позволяет определить цвет по трем составляющим. Так, команда SetPenColor (clRed); аналогична команде SetPenColor (RGB(255,0,0)); Такой способ задания цвета позволяет задавать цвета, значения которых не описаны цветовыми константами. Значения составляющих цвета можно посмотреть в графическом редакторе Paint (пример 14.9).

В графическом режиме PascalABC можно выводить в графическое окно тексты и числа.

TextOut(x,y,z); — выводит строку или число z в прямоугольник с координатами левого верхнего угла (x,y).

#### Пример 14.8. Программа:

```
uses GraphABC;
begin
  //Дом
  SetPenColor(RGB(255,0,0));
  SetBrushColor(clBlue);
  Rectangle(100,200,400,450);
  //Окно
  SetBrushColor(clYellow);
  Rectangle(200,250,300,350);
  SetPenColor(clRed);
  SetPenStyle(psDot);
  SetPenWidth(2);
  Line(250,250,250,350);
  Line(200,300,300,300);
  //Крыша
  SetPenStyle(psSolid);
  SetPenWidth(1);
  Line(100, 200, 250, 0);
  Line(250, 0, 400, 200);
  SetBrushStyle(bsHatch);
  SetBrushColor(clLightGreen);
  SetBrushHatch(bhCross);
  Circle(250, 125, 30);
  FloodFill(250,70, clPlum);
end.
```

Результат работы программы:



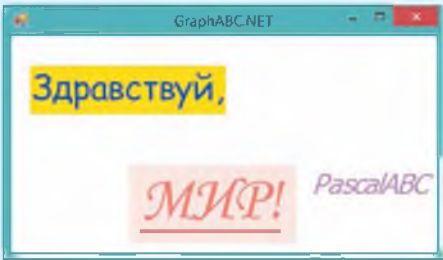
#### Пример 14.9. Составляющие цвета:



**Пример 14.10.** Выведем в графическое окно приветствие миру, используя разные свойства текста.

```
Программа:
uses GraphABC;
begin
  //Цвет фона для текста
  SetBrushColor(clYellow);
  SetFontName('Comic Sans MS');
  //Цвет букв
  SetFontColor(clBlue);
  //Размер шрифта
  SetFontSize(25);
  //Полужирный шрифт
  SetFontStyle(fsBold);
  TextOut(20,30,'Здравствуй,');
  SetBrushColor(clPink);
  StFontName('Monotype Corsiva');
  SetFontColor(clSalmon);
  SetFontSize(50);
  SetFontStyle(fsUnderline);
  TextOut(120,130,'МИР!');
  //Прозрачный фон
  SetBrushStyle(bsClear);
  SetFontName('Tahoma');
  SetFontColor(clViolet);
  SetFontSize(20);
  SetFontStyle(fsItalic);
  TextOut(300,130,'PascalABC');
end.
```

Результат выполнения программы:



Если нужно вывести строку, то ее символы заключают в кавычки, для вывода числа можно использовать переменные или значения чисел. Если в качестве *z* записать арифметическое выражение, то будет выведено его значение.

Для изменения параметров текста применяют следующие команды:

Команда	Описание
SetFontColor	Цвет символов
SetFontSize	Размер символов
SetFontName	Имя текущего шрифта
SetFontStyle	Стиль текста

Команда `SetBrushColor` устанавливает цвет прямоугольника, внутри которого будет находиться текст (пример 14.10). Команда действует до тех пор, пока цвет не будет изменен. Для записи текста без фона нужно установить прозрачный фон кисти: `SetBrushStyle(bsClear)`.

Все параметры для текста задаются до команды вывода его в графическое окно. Имя шрифта, которым будет выведен текст, заключается в кавычки. Возможные варианты можно посмотреть в Word. Стиль текста может иметь следующие значения: `fsNormal` (обычный), `fsBold` (жирный), `fsItalic` (наклонный), `fsUnderline` (подчеркнутый) или их комбинации: `fsBoldUnderline` (жирный подчеркнутый).

- 
1. Какая библиотека используется для подключения графики в PascalABC?
  2. Как определена система координат в графическом окне?
  3. Как задать размеры графического окна?
  4. Как найти описание графических примитивов в справочнике?
  5. Как можно изменить цвет линий, заливки?
  6. Как вывести текст в графическом окне?





## Упражнения

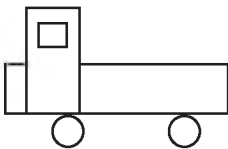
- 1 Подпишите изображение из примера 14.8.
- 2 Дополните изображение домика из примера 14.8 изображениями трубы и дыма из трубы в виде нескольких овалов:
- 3 Дополните результат, полученный при выполнении задания 2, какими-либо из предложенных изображений или придумайте свои.



- 4 Напишите программу для создания изображения. Раскрасьте данное изображение по своему усмотрению. Дополнительные команды для построения графических примитивов можно найти в справочной системе.

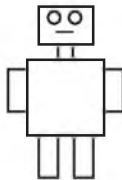
Грузовик

1



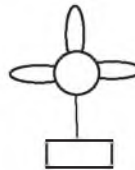
Робот

2



Цветок

3



Чебурашка

4



## § 15. Простые и составные условия

### 15.1. Логический тип данных

Напомним изученные в 7-м классе понятия *высказывание* и *условие для исполнителя*.

**Высказывание** — повествовательное предложение (утверждение), о котором можно сказать, истинно оно или ложно.

**Условием для исполнителя** является известное ему высказывание, которое может соблюдаться (быть

Тип Boolean назван в честь английского математика и логика Джорджа Буля, занимавшегося вопросами математической логики в XIX в.

Данный тип присутствует в подавляющем большинстве языков программирования. В некоторых языках реализуется через числовой тип данных. Тогда за значение ложь принимается 0, а за значение истина — 1.

**Пример 15.1.** Примеры логических выражений:

- $3 < 7$  — логическое выражение, значение которого true;
- $2 + 2 * 2 = 8$  — логическое выражение, значение которого false;
- $\text{abs}(-5) > \text{abs}(3)$  — логическое выражение, значение которого true;
- $y \geq \text{sqr}(x)$  — логическое выражение, значение которого можно определить, только зная значения переменных  $x$  и  $y$ . При  $x = 2$  и  $y = 10$  значение выражения — true. При  $x = 10$  и  $y = 2$  — false.

Проверим истинность этих выражений в программе:

```
var a1, a2, a3, a4, a5: boolean;  
    x, y: integer;  
begin  
    a1 := 3 < 7;  
    writeln('a1 =', a1);  
    a2 := 2 + 2 * 2 = 8;  
    writeln('a2 =', a2);  
    a3 := abs(-5) > abs(3);  
    writeln('a3 =', a3);  
    x := 2; y := 10;  
    a4 := y >= sqr(x);  
    writeln('a4 = ', a4);  
    x := 10; y := 2;  
    a5 := y >= sqr(x);  
    writeln('a5 =', a5);  
end.
```

Результат работы программы:

Окно вывода	
a1 =	True
a2 =	False
a3 =	True
a4 =	True
a5 =	False

По умолчанию  $\text{false} < \text{true}$ .

истинным) либо не соблюдаться (быть ложным).

В языке программирования Pascal для работы с условиями определен логический тип данных **boolean**. Величины типа **boolean** могут принимать два значения — **false** (ложь) и **true** (истина).

Значения **false** и **true** получаются в результате выполнения операций сравнения над числовыми данными. Для сравнения используют знаки, указанные в таблице.

Операция	PascalABC
Равно (=)	=
Не равно ( $\neq$ )	$\langle \rangle$
Больше ( $>$ )	$>$
Меньше ( $<$ )	$<$
Больше или равно ( $\geq$ )	$\geq$
Меньше или равно ( $\leq$ )	$\leq$

Сравнить можно константы, переменные, арифметические и логические выражения.

**Логическое выражение** — выражение, принимающее одно из двух значений: **true** или **false**.

Логические выражения можно присваивать переменным типа **boolean**, а также выводить их значения на экран: будет выведено слово **false** или **true** соответственно (пример 15.1). Условия для исполнителя являются частным случаем логических выражений.

**Пример 15.2.** Написать программу, которая выведет на экран значение **true** или **false** в зависимости от того, является ли введенное число  $x$  четным или нет.

Этапы выполнения задания

I. Исходные данные:  $x$  (введенное число).

II. Результат:  $a$  (true или false).

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Вычисление значения логической переменной. Число является четным, если остаток от деления его на 2 равен нулю. Значение переменной  $a$  определяется значением выражения  $x \bmod 2 = 0$ .

3. Вывод результата.

IV. Описание переменных:  $x$  — integer,  $a$  — boolean.

## 15.2. Составные условия

С высказываниями можно производить логические операции (**НЕ**, **И**, **ИЛИ**). Для логических переменных также определены логические операции, соответствующие операциям над высказываниями: **not**, **and**, **or**.

Логические выражения, в которых наряду с простыми условиями (сравнениями) используются логические операции, называют **составными условиями**.

Приведем таблицы истинности логических операций.

Логическая переменная		Результат операции		
$A$	$B$	<b>not</b> $A$	$A$ <b>and</b> $B$	$A$ <b>or</b> $B$
True	True	False	True	True
False	True	True	False	True
True	False	False	False	True
False	False	True	False	False

### Пример 15.2.

V. Программа:

```
var x: integer;
    a: boolean;
begin
  write('Введите x = ');
  read(x);
  a := x mod 2 = 0;
  write('Число четное - ', a);
end.
```

VI. Тестирование

Запустить программу и ввести значение  $x = 6$ . Результат:

#### Окно вывода

```
Введите x = 6
Число четное - True
```

Запустить программу и ввести значения  $x = 11$ . Результат:

#### Окно вывода

```
Введите x = 11
Число четное - False
```

В языке PascalABC реализована логическая операция **xor** — исключающее **ИЛИ**. Этой операции соответствует высказывание: «Только одно из двух высказываний может быть истинно». Таблица истинности для операции **xor**:

$A$	$B$	$A$ <b>xor</b> $B$
True	True	False
False	True	True
True	False	True
False	False	False

Все логические операции могут применяться к числам типа integer. Число рассматривается в двоичном представлении, и операции применяются к битам числа. Бит, равный 1, представляется как истина, а бит, равный нулю, — как ложь.

**Пример 15.3.** Определение порядка действий для выражения ( $a, d$  — boolean,  $c, b$  — integer):

$a \text{ or } (c < b) \text{ and } d$

Первым выполняется сравнение  $c$  и  $b$ , затем логическая операция **and**, потом — **or**.

**Пример 15.4\*.** Рассмотрим выражение:

$a < b \text{ and } c < d$

Если  $a, b, c, d$  имеет тип integer, то получим ошибку: «Операция ' $<$ ' не применима к типам boolean и integer» (с помощью знака ' $<$ ' нельзя сравнивать число и логическую переменную). Если переменные имеют тип real, то возникнет ошибка: «Операция '**and**' не применима к типу real». Правильная запись выражения:

$(a < b) \text{ and } (c < d)$

Все вышеперечисленные ошибки возникают потому, что операция **and** обладает большим приоритетом по отношению к операциям  $<$ . Поэтому сначала будет производиться попытка выполнить операцию  $b \text{ and } c$ , а затем сравнения.

**Пример 15.5.** Построение отрицаний:

**not not**  $a = a$ ;  
**not** ( $a \text{ and } b$ ) = (**not**  $a$ ) **or** (**not**  $b$ );  
**not** ( $a \text{ or } b$ ) = (**not**  $a$ ) **and** (**not**  $b$ ).

Рассмотрим выражение **not**  $a < b$  с переменными  $a$  и  $b$  типа integer. Здесь операция **not** относится к переменной  $a$ , поэтому в двоичном представлении числа  $a$  биты, равные 1, будут заменены на 0, а биты, равные 0, — на 1. Затем полученный результат сравнится с числом  $b$ . Для отрицания сравнения выражение нужно записать так: **not** ( $a < b$ ).

В логических выражениях могут встречаться как арифметические операции, так и логические. Порядок выполнения операций определяется их приоритетом:

- 1) **not**;
- 2)  $*$ ,  $/$ , **div**, **mod**, **and**;
- 3)  $+$ ,  $-$ , **or**;
- 4)  $=$ ,  $<>$ ,  $<$ ,  $>$ ,  $<=$ ,  $>=$ .

(Рассмотрите пример 15.3.)

Операции с одинаковым приоритетом выполняются по порядку, слева направо. Для изменения порядка выполнения операций применяют скобки (пример 15.4).

При составлении программ часто нужно строить отрицания сложным логическим выражениям. Для этого полезно использовать тождества, известные из алгебры логики (пример 15.5), и следующую таблицу:

Условие	Противоположное условие (отрицание условия)
$a < b$	$a >= b$
$a > b$	$a <= b$
$a = b$	$a <> b$

**Пример 15.6.** Написать программу, которая выдаст на экран значение **true** или **false** в зависимости от того, находится ли число  $B$  между числами  $A$  и  $C$ .

Этапы выполнения задания

I. Исходные данные: переменные  $A, B, C$  (вводимые числа).

II. Результат: **rez** (True или False).

III. Алгоритм решения задачи.

1. Ввод исходных данных.



2. Вычисление значения логической переменной. Рассмотрим два случая.

Верно неравенство:  $A < B < C$ . Этому неравенству соответствует логическое выражение:  $(A < B) \text{ and } (B < C)$ . При своем переменной `r1` значение этого выражения.

Верно неравенство:  $A > B > C$ . Этому неравенству соответствует логическое выражение:  $(A > B) \text{ and } (B > C)$ . При своем переменной `r2` значение этого выражения.

Ответом на задачу будет значение логического выражения `r1 or r2`.

3. Вывод результата.

IV. Описание переменных: `A, B, C` — `integer`, `r1, r2, rez` — `boolean`.

Для работы с логическими величинами могут использоваться функции. Функция `Ord` (порядковый номер значения) позволяет преобразовать логическое значение в числовое: `Ord(false) = 0`, а `Ord(true) = 1`. Функции `Pred` (предшествующее значение) и `Succ` (последующее значение) позволяют преобразовывать логические значения:

`D := Pred(true); {D = false}`

`E := Succ(false); {E = true}`



1. Что такое составное условие?
2. Назовите логические операции, используемые в PascalABC.
3. Какой приоритет у логической операции **not** (**and**, **or**)?



## Упражнения

- 1 Сформулируйте и реализуйте обратную задачу для примера 15.2: для всех тех случаев, для которых в исходной задаче было `true`, нужно вывести `false` и, наоборот, для всех тех случаев, в которых в исходной задаче получалось `false`, получить `true`.
- 2 В PascalABC определена логическая функция `odd(x)`. Значение этой функции `true`, если число `x` является нечетным, и `false`, если `x` — четное. Измените программу примера 15.2, используя функцию `odd`.

### Пример 15.6.

V. Программа:

```
var A, B, C: integer;
    r1, r2, rez: boolean;
begin
    writeln('Введите A, B, C');
    read(A, B, C);
    r1 := (A < B) and (B < C);
    r2 := (A > B) and (B > C);
    rez := r1 or r2;
    write('Число B между числами
           A и C — ', rez);
end.
```

VI. Тестирование.

Запустить программу и ввести значения `A = 5, B = 0, C = -5`. Результат:

#### Окно вывода

Введите A, B, C

5 0 -5

Число B между числами A и C - True

Запустить программу и ввести значения `A = -2, B = -7, C = 5`. Результат:

#### Окно вывода

Введите A, B, C

-2 -7 5

Число B между числами A и C - False


VII. Анализ результатов. Для полного тестирования программы нужно проверить все возможные случаи взаимного расположения `A, B, C` (их всего 6).

- 3 Определите, что делают следующие программы, и дополните команду вывода.
1. **var** x: integer;  
a: boolean;  
**begin**  
write('Введите x = ');  
read(x);  
a := x mod 10 = 0;  
write('Число ... — ', a);  
**end.**
  2. **var** x: integer;  
a: boolean;  
**begin**  
write('Введите x = ');  
read(x);  
a := (x > 10) **and** (x < 100);  
write('Число ... — ', a);  
**end.**
- 4 Напишите программу, которая выведет на экран значение true или false, в зависимости от того, является ли введенное число x положительным или нет.
- 5 Напишите программу, которая выведет на экран значение true или false, в зависимости от того, является ли введенное число x четырехзначным или нет.
- 6\* Заданы два положительных числа x и y. Определите, верно ли, что первое число меньше второго и хотя бы одно из них нечетное. Выведите на экран true или false.

## § 16. Оператор ветвления

Использование управляющих конструкций предполагает запись программы в структурированном виде. Структурированность программ достигается за счет отступов, регулирующих размещение вложенных алгоритмических конструкций.

Можно соблюдать следующее правило: при движении курсора вниз от «начала» структуры до ее «конца» на пути курсора могут встретиться только пробелы. Все, что находится «внутри» структуры, размещается правее.

Кнопка  позволяет преобразовать код программы к структурированному виду.

### Пример 16.1.

V. Программа:

```
var x: integer;  
begin  
write('Введите x = '); read(x);  
if x > 0 then  
    write('положительное')  
else  
    write('не положительное');  
end.
```

### 16.1. Запись оператора ветвления

Алгоритмическая конструкция *ветвление* (см. блок-схему в примере 13.2, с. 60) обеспечивает выполнение одной или другой последовательности команд в зависимости от истинности или ложности некоторого условия.

**Оператор ветвления** — команда, реализующая алгоритмическую конструкцию *ветвление* на языке программирования.

Для записи оператора ветвления используют команды **if**. Формат команды:

```
if <условие> then
```

```
begin
```

```
    Команды 1;
```

```
end
```

```
else
```

```
begin
```

```
    Команды 2;
```

```
end;
```

Оператор ветвления может быть в полной или в сокращенной форме. В сокращенной форме отсутствует блок `else`:

```
if <условие> then
begin
  Команды;
end;
```

Условие в записи оператора ветвления бывает простым и составным. Операторные скобки могут быть опущены, если внутри их находится одна команда.

**Пример 16.1.** Задано число  $x$ . Нужно определить, является ли оно положительным или нет, и вывести соответствующее сообщение.

Этапы выполнения задания

I. Исходные данные:  $x$  (введенное число).

II. Результат: соответствующее сообщение.

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Проверка значения выражения ( $x > 0$ ).

3. Вывод результата.

IV. Описание переменных:  $x$  — integer.

## 16.2. Решение задач с использованием оператора ветвления

**Пример 16.2.** В момент времени 00:00 на светофоре для пешеходов включили зеленый сигнал. Далее сигнал светофора сменяется каждую минуту: 1 минуту горит зеленый сигнал, 1 минуту — красный. Известно, что с момента включения светофора прошло

### Пример 16.1. Продолжение.

VI. Тестирование.

Запустить программу и ввести значение  $x = 5$ . Результат:

#### Окно вывода

Введите  $x = 5$   
положительное

Запустить программу и ввести значение  $x = -1$ . Результат:

#### Окно вывода

Введите время  $x = -1$   
не положительное

VII. Анализ результатов. Для полной проверки программы требуется еще проверить значение  $x = 0$ .

#### Окно вывода

Введите  $x = 0$   
не положительное

### Пример 16.2.

V. Программа:

```
uses GraphABC;
var m:integer;
begin
  Rectangle(250,50,390,250);
  SetBrushColor(clBlack);
  Circle(320,100,30);
  Circle(320,200,30);
  SetBrushColor(clWhite);
  writeln('Введите время');
  read(m);
  writeln(m)1;
  if m mod 2 = 1 then
    FloodFill(320,100,clRed)
  else
    FloodFill(320,200,clGreen);
end.
```

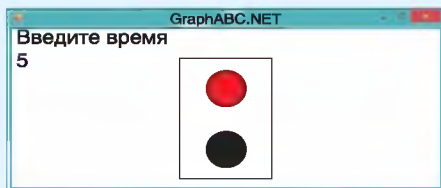
<sup>1</sup> При вводе данных в графическом окне они не отображаются на экране. Для того чтобы видеть, что ввели, необходимо дополнительно вывести введенное значение.

**Пример 16.2. Продолжение.****VI. Тестирование.**

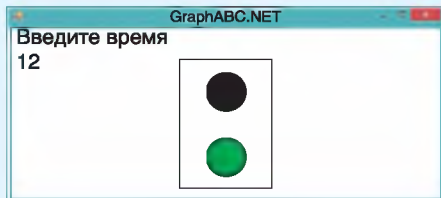
Вид графического окна до ввода числа:



Ввести значение  $x = 5$ . Результат:



Ввести значение  $x = 12$ . Результат:

**Пример 16.3.****V. Программа:**

```

Var x1, y1, x2, y2, r_T, r_K: real;
begin
  writeln('Танин дом'); read(x1,y1);
  writeln('Катин дом'); read(x2,y2);
  r_T := sqrt(x1*x1+y1*y1);
  r_K := sqrt(x2*x2+y2*y2);
  if r_T < r_K then
    writeln('Танин дом ближе')
  else
    writeln('Катин дом ближе');
  end.

```

**VI.** Запустить программу и ввести значения: Танин дом —  $x1 = 2.3$ ,  $y1 = 4.5$ , Катин дом —  $x2 = -2.1$ ,  $y2 = 4.9$

$m$  минут. Требуется нарисовать светофор с включенным сигналом в соответствии с введенным значением времени.

Этапы выполнения задания

I. Исходные данные:  $m$  (заданное время).

II. Результат: рисунок светофора, зависящий от значения  $m$ .

III. Алгоритм решения задачи.

1. Рисование светофора (прямоугольник и 2 круга) с выключенными сигналами.

2. Ввод исходных данных.

3. Цвет сигнала будет зависеть от того, четным или нечетным будет значение  $m$ . Если  $m$  четное — сигнал зеленый (закрашиваем нижний круг), если нечетное — красный (закрашиваем верхний круг).

4. Закрасим нужный круг цветом в зависимости от четности  $m$ .

IV. Описание переменных:  $m$  — integer.

**Пример 16.3.** Таня и Катя живут в разных домах. Им стало интересно, кто из них живет ближе к школе. Они разместили на карте прямоугольную систему координат так, чтобы школа имела координаты  $(0; 0)$ . Известно, что Танин дом имеет координаты  $(x1; y1)$ , а Катин  $(x2; y2)$ . Девочки ходят в школу по прямой и проходят разные расстояния. Нужно написать программу, которая определит, чей дом ближе к школе.

Этапы выполнения задания

I. Исходные данные: координаты домов девочек  $x1, y1, x2, y2$ .

II. Результат: сообщение о том, чей дом ближе.

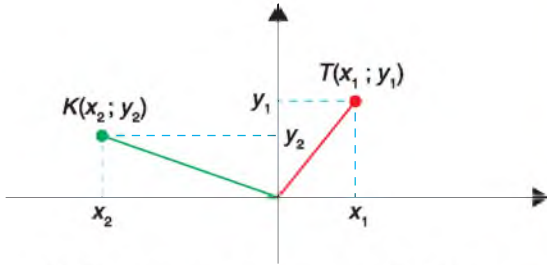
III. Алгоритм решения задачи.



1. Ввод координат домов.

2. Вычисление расстояний до школы:  $r_T$  (от Таниного дома) и  $r_K$  (от Катиного дома). Для вычисления воспользуемся теоремой Пифагора:

$$r_T = \sqrt{x_1^2 + y_1^2} \text{ и } r_K = \sqrt{x_2^2 + y_2^2}.$$



3. Сравнение расстояний. Вывод ответа.

IV. Описание переменных:  $x1, y1, x2, y2, r_T, r_K$  имеют тип `real`.

**Пример 16.4.** Вася начал заниматься стрельбой из лука. Для тренировки он решил создать модель мишени, которая будет реагировать на лазер. Мишень представляет собой два круга (стреляет Вася пока не очень хорошо) разного радиуса с общим центром. Если Вася попал в маленький круг, то круг загорается зеленым. Большой круг при попадании в него загорается желтым. Если Вася не попал ни в один из кругов, то область вне кругов загорается красным. Необходимо реализовать компьютерную модель Васиной мишени (при попадании на границу круга ничего не должно происходить).

Этапы выполнения задания

I. Исходные данные: координаты точки выстрела ( $x; y$ ).

II. Результат: рисунок мишени.

III. Алгоритм решения задачи.

**Пример 16.3. Продолжение.**

Результат должен быть таким:

#### Окно вывода

```
Танин дом
2.3 4.5
Катин дом
-2.1 4.9
Танин дом ближе
```

**Пример 16.4.**

V. Программа:

```
uses GraphABC;
var x,y, x0, y0, R_b, R_m, z:
integer;
begin
  x0 := 320; y0 := 240;
  R_b := 150; R_m := 75;
  Circle(x0,y0,R_b);
  Circle(x0,y0,R_m);
  writeln('Выстрел');
  read(x,y);
  writeln(x, ' ', y);
  z := sqr(x-x0)+sqr(y-y0);
  if z < sqr(R_m) then
    FloodFill(x,y,clLightGreen)
  else
    if z < sqr(R_b) then
      FloodFill(x,y,clYellow)
    else
      FloodFill(x,y,clRed);
end.
```

VI. Тестирование.

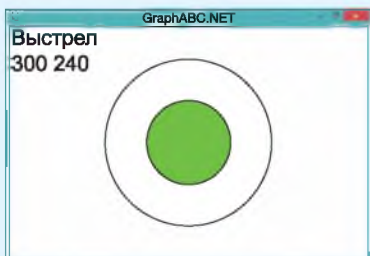
Запустить программу и ввести координаты выстрела (240; 240).

Результат:

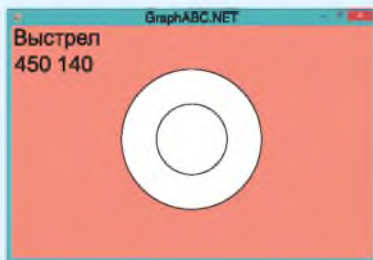


**Пример 16.4. Продолжение.**

Запустить программу еще раз и ввести координаты выстрела (300; 240).



Запустить программу еще раз и ввести координаты выстрела (450; 140).

**Пример 16.5.**

V. Программа:

```
var a, a1, a2, a3: integer;
begin
  write('Введите a = ');
  read(a);
  if (a > 99) and (a < 1000) then
    begin
      //Первая цифра
      a1 := a div 100;
      //Вторая цифра
      a2 := a mod 100 div 10;
      //Третья цифра
      a3 := a mod 10;
      writeln(a1);
      writeln(a2);
      writeln(a3);
    end
  else
    writeln('не трехзначное');
  end.
```

1. Рисование мишени: 2 круга радиусов  $R_b = 150$  и  $R_m = 75$  с центром в точке  $(x_0; y_0)$ ,  $x_0 = 320$ ,  $y_0 = 240$ . Сначала рисуем круг большего радиуса.

2. Ввод данных: координаты точки выстрела.

3. Цвет рисунка будет зависеть от того, в какую область относительно кругов попала точка. Возможны 3 случая:

1) точка внутри маленького круга. Длина отрезка между точкой и центром круга меньше радиуса. По теореме Пифагора:

$$(x - x_0)^2 + (y - y_0)^2 < R_m^2;$$

2) если условие а) не выполняется, проверяем, принадлежит ли точка большому кругу:

$$(x - x_0)^2 + (y - y_0)^2 < R_b^2;$$

3) если условия а) и б) не выполняются, то Вася не попал в мишень.

4. Для сокращения записи определим переменную  $z = (x - x_0)^2 + (y - y_0)^2$ .

5. Закрасим нужную область цветом в зависимости от проверки условий.

IV. Описание переменных:  $x$ ,  $y$ ,  $x_0$ ,  $y_0$ ,  $R_b$ ,  $R_m$ ,  $z$  имеют тип integer.

**Пример 16.5.** Проверить, является ли введенное число трехзначным, и если да, то вывести цифры этого числа в отдельных строках.

Этапы выполнения задания

I. Исходные данные:  $a$  (трехзначное число).

II. Результат: переменные  $a1$ ,  $a2$ ,  $a3$  (цифры числа) или сообщение «не трехзначное».

III. Алгоритм решения задачи.

1. Ввод исходного числа.

2. Проверка числа. Число  $a$  является трехзначным, если  $99 < a < 1000$ .

3. Если число трехзначное, выделяем его цифры:

1) для выделения первой цифры  $a_1$  находим целую часть от деления числа  $a$  на 100;

2) для выделения второй цифры  $a_2$  числа  $a$  находим остаток от его деления на 100, а затем целую часть от деления полученного остатка на 10;

3) последняя цифра числа  $a_3$  является остатком от деления числа  $a$  на 10.

4. Вывод результата.

IV. Описание переменных: все переменные имеют тип `integer`.

#### Пример 16.5. Продолжение.

##### VI. Тестирование.

Запустить программу и ввести значение 345.

Результат следующий:

##### Окно вывода

Введите  $a = 345$

3

4

5

Другой вариант исходных данных:

##### Окно вывода

Введите  $a = 24$

не трехзначное



1. Что такое оператор ветвления?
2. Чем отличается полная запись оператора ветвления от сокращенной?
3. Можно ли использовать составные условия в операторе ветвления?



### Упражнения

- 1 Можно ли изменить логическое выражение в операторе ветвления в примере 16.1 так, чтобы сообщения 'положительное' и 'не положительное' пришлось поменять местами? Если да, то как это сделать?
- 2\* Какие изменения нужно внести в программу примера 16.1, чтобы для числа рассматривались три случая: 'положительное', 'отрицательное', 'равно нулю'?
- 3 Подключите графический режим в программе примера 16.1. Измените программу так, чтобы сообщение 'положительное' выводилось красным цветом, а сообщение 'не положительное' — синим.
- 4\* Измените программу в примере 16.2 так, чтобы четность (нечетность) числа проверялась с использованием функции `odd`.
- 5 Напишите программу. Задано число  $x$ . Если число четное, то нарисовать на экране зеленый прямоугольник, а если нечетное, то красный круг (см. пример 16.2).
- 6 Добавьте в программу из примера 16.3 проверку корректности исходных данных: координаты домов должны быть такими, чтобы расстояния до школы были разными. Если расстояния одинаковы, то вывести сообщение 'Координаты введены неверно', а если разные, то вывести ответ.
- 7 Какие изменения понадобится внести в программу из примера 16.3, если допустить, что девочки могут проходить одинаковые расстояния? Внесите изменения в программу и проверьте правильность ее работы.

8 Для усложнения тренировок Вася (пример 16.4) решил менять местоположение мишени и радиусы кругов. Добавьте в программу возможность ввода радиусов большого и маленького кругов, а также центра мишени. Проверьте правильность работы программы на различных наборах исходных данных.

9 Как известно, многие задачи имеют не единственное решение. Так, Юлия нашла другой способ вычисления второй цифры трехзначного числа для примера 16.5. Какую из команд использовала Юлия? Объясните, что получится при выполнении каждой из приведенных команд.

1) `a2:=a mod 10 div 10;`      2) `a2:=a div 10 mod 10;`      3) `a2:=a div 100 mod 10.`

10 Программу из примера 16.5 изменили. Сформулируйте условие задачи, которая решается с помощью этой программы.

```
var a, a1, a2, a3: integer;
begin
  write('Введите a = '); read(a);
  if (a > 99) and (a < 1000) then
    begin
      // Первая цифра
      a1 := a div 100;
      // Вторая цифра
      a2 := a mod 100 div 10;
      // Третья цифра
      a3 := a mod 10;
      if a1 mod 2 = 0 then
        writeln(a1, '— четная ');
      if a2 mod 2 = 0 then
        writeln(a2, '— четная ');
      if a3 mod 2 = 0 then
        writeln(a3, '— четная');
      if odd(a1) and odd(a2) and odd(a3) then
        writeln('нет четных цифр');
    end
  else
    writeln('не трехзначное');
  end.
```

11 Программу из задания 10 проверили для некоторых случаев. Все ли возможные ситуации рассмотрели? Что нужно добавить?

<div>Окно вывода</div> <div>Введите a = 246</div> <div>2 - четная</div> <div>4 - четная</div> <div>6 - четная</div>	<div>Окно вывода</div> <div>Введите a = 103</div> <div>0 - четная</div>	<div>Окно вывода</div> <div>Введите a = 537</div> <div>нет четных цифр</div>	<div>Окно вывода</div> <div>Введите a = 26</div> <div>не трехзначное</div>
---	---	--	--



**12** Петя решил усовершенствовать программу из задания 10 и проверку цифр в числе записал следующим образом:

```

if a1 mod 2 = 0 then
    writeln(a1, ' — четная')
else
    if a2 mod 2 = 0 then
        writeln(a2, ' — четная')
    else
        if a3 mod 2 = 0 then
            writeln(a3, ' — четная')
        else
            writeln('нет четных цифр');

```

Почему Петина отметка оказалась невысокой? Приведите примеры, для которых программа выдает неправильный ответ. Приведите примеры, когда программа выдает правильный ответ, если такое возможно.

**13** Дано натуральное число. Напишите программу, которая проверяет, является ли оно трехзначным и кратна ли 7 сумма его цифр.

**14\*** Дано натуральное число. Напишите программу, которая проверяет, является ли оно четырехзначным и расположены ли его цифры в порядке убывания.

**15\*** Вася научился попадать в центр мишени из примера 16.4 и решил перейти к более сложным тренировкам. Теперь его мишень представляет собой три вложенных круга с радиусами  $R_1$ ,  $R_2$ ,  $R_3$  (известно, что  $R_1 < R_2 < R_3$ ). Реализуйте компьютерную модель этой мишени. Цвета выберите самостоятельно.

## § 17. Оператор цикла

### 17.1. Оператор цикла с предусловием

Алгоритмическая конструкция *повторение (цикл)* представляет собой последовательность действий, выполняемых многократно (см. блок-схему в примере 13.2, с. 60). Саму последовательность называют **телом цикла**.

**Оператор цикла** — команда, реализующая алгоритмическую конструкцию *повторение* на языке программирования.

В Pascal существуют разные возможности управлять тем, сколько раз будет повторяться тело цикла. Может быть задано условие продолжения или

Цикл с заданным условием окончания работы в PascalABC записывается следующим образом:

```

repeat
    тело цикла;
until <условие>;

```

Цикл работает, пока условие ложно, и прекращает работу, когда условие становится истинным.

Этот цикл называют **циклом с постусловием**, так как проверка условия осуществляется после выполнения тела цикла. Цикл с постусловием всегда выполняется хотя бы один раз.

Циклы **repeat** и **while** в PascalABC взаимозаменяемы, поэтому при написании программ достаточно использования только одного из них.

**Пример 17.1.**

V. Программа:

```

uses GraphABC;
var x, y, r: integer;
begin
  r := 10;
  x := 10; y := 10;
  while x < 640 do
    begin
      Circle(x, y, r);
      x := x + 20;
    end;
  end.

```

VI. Тестирование

Запустить программу. Результат:



VII. Числовое значение в условии цикла можно заменить функцией, определяющей горизонтальное разрешение окна: `WindowWidth`:

`while x < WindowWidth do`  
 Функции `RedColor`, `GreenColor`, `BlueColor` позволяют менять интенсивность соответствующего цвета.

```

uses GraphABC;
var x, y, r, c: integer;
begin
  r := 10;
  x := 10; y := 10;
  c := 255;
  while x < 640 do
    begin
      //Интенсивность красного
      SetBrushColor(RedColor(c));
      Circle(x,y,r);
      x := x + 20;
      //Уменьшение интенсивности
      c := c-5;
    end;
  end.

```



окончания работы цикла, а также число повторений тела цикла.

**Цикл с предусловием** используется в том случае, когда известно условие продолжения работы. Для записи оператора цикла с предусловием используется команда **while**. Формат команды:

```

while <условие> do
begin
  тело цикла;
end;

```

**Пример 17.1.** Написать программу для рисования ряда окружностей с радиусом 10 пикселей вдоль верхнего края графического окна.

Этапы выполнения задания

I—II. Результатом работы программы, не зависящей от исходных данных, будет рисунок, отображающий ряд окружностей вдоль верхнего края графического окна.

III. Алгоритм решения задачи.

1. *Положение первой окружности.*

Окружность расположим в верхнем левом углу. Для этого задается радиус  $r = 10$  и координаты центра  $x = 10$ ,  $y = 10$ .

2. *Положение любой другой окружности*, удовлетворяющей условию задачи, будет зависеть от координаты  $x$ . В цикле будем изменять значение  $x$ . Каждое новое значение будет на 20 (на размер диаметра) больше предыдущего.

3. Цикл должен завершиться, когда значение координаты  $x$  станет больше чем 640 — горизонтальный размер окна.

IV. Описание переменных:  $x$ ,  $y$ ,  $r$  — integer.

## 17.2. Оператор цикла с параметром

Цикл с параметром используется тогда, когда известно количество повторений.

Для записи оператора цикла с параметром используется команда `for`. Формат команды:

```
for var1 i := N1 to N2 do
begin
    тело цикла;
end;
Или
for var i := N2 downto N1 do
begin
    тело цикла;
end;
```

В первом варианте параметр цикла  $i$  изменяется от  $N1$  до  $N2$ , каждый раз увеличиваясь на 1. Во втором — параметр  $i$  уменьшается на 1 при каждом выполнении тела цикла от  $N2$  до  $N1$ . Если  $N1 > N2$ , цикл не выполняется ни разу. Изменять значение параметра внутри тела цикла нельзя.

**Пример 17.2.** Написать программу для вывода таблицы умножения на заданное число  $x$ .

Этапы выполнения задания

I. Исходные данные:  $x$  (введенное число).

II. Результат: 9 строк вида  $a * x = c$ .

III. Алгоритм решения задачи.

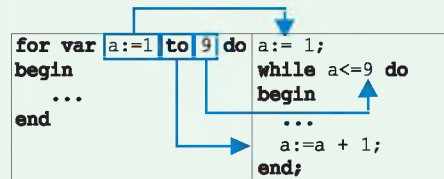
1. Значение переменной  $a$  изменяется в цикле от 1 до 9.

2. Значение переменной  $c = a \cdot x$ .

3. Так как количество повторений заранее известно, используем цикл `for`.

IV. Описание переменных:  $x$ ,  $c$  — `integer`.

Любой цикл `for` может быть заменен на цикл `while`:



Обратное не всегда возможно.

### Пример 17.2.

V. Программа:

```
var x, c : integer;
begin
    write('Введите x = '); read(x);
    for var a := 1 to 9 do
    begin
        c := a * x;
        writeln(a, ' * ', x, ' = ', c);
    end;
end.
```

VI. Тестирование.

Запустить программу. Ввести  $x = 7$ .

#### Окно вывода

```
Введите x = 7
1 * 7 = 7
2 * 7 = 14
3 * 7 = 21
4 * 7 = 28
5 * 7 = 35
6 * 7 = 42
7 * 7 = 49
8 * 7 = 56
9 * 7 = 63
```

Решение с помощью цикла `while`:

```
var a, x, c : integer;
begin
    write('Введите x = '); read(x);
    a := 1;
    while a <= 9 do
    begin
        c := a * x;
        writeln(a, ' * ', x, ' = ', c);
        a := a + 1;
    end;
end.
```

VII. Проверить результат.

<sup>1</sup> Ключевое слово `var` может быть опущено, тогда переменная  $i$  должна быть описана (как `integer`) в разделе описания `var` до начала программы.

При решении задач с использованием оператора цикла важно правильно выбрать вид цикла. Если известно количество повторений тела цикла, то выбирают цикл **for**, а иначе — цикл **while**.

Внутри цикла можно использовать операторы **break** (немедленный выход из текущего цикла) и оператор **continue** (переход к концу тела цикла).

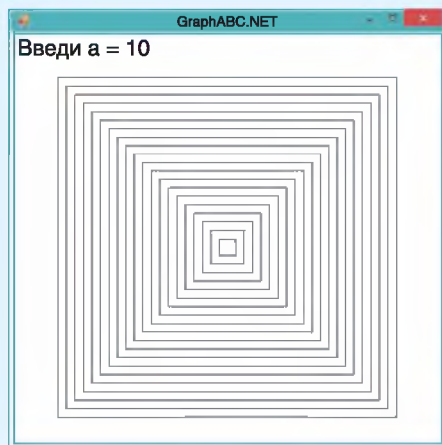
#### Пример 17.3.

V. Программа:

```
uses GraphABC;
var a,x1,y1,x2,y2: integer;
begin
  write('Введи a = ');
  read(a); write(a);
  x1 := 50; y1 := 50;
  x2 := 450; y2 := 450;
  for var i := 1 to 20 do
  begin
    Rectangle(x1,y1, x2,y2);
    x1 := x1 + a;  y1 := y1 + a;
    x2 := x2 - a;  y2 := y2 - a;
  end;
end.
```

VI. Тестирование.

Запустить программу и ввести значение  $a = 10$ . Результат:



### 17.3. Решение задач

#### с использованием оператора цикла

**Пример 17.3.** Нарисовать 20 квадратов с общим центром. Длина стороны самого большого квадрата 400, верхний левый угол расположен в точке (50; 50). Координаты верхнего левого и нижнего правого углов каждого следующего квадрата изменяются на  $a$  ( $a$  — вводится).

Этапы выполнения задания

I. Исходные данные:  $a$  (введенное число).

II. Результат: рисунок, отображающий квадраты.

III. Алгоритм решения задачи.

1. Первым рисуется самый большой квадрат. Координаты его верхнего левого угла  $x1 = 50$ ,  $y1 = 50$ . Координаты нижнего правого угла  $x2 = 450$ ,  $y2 = 450$ .

2. Для определения положения другого квадрата нужно координаты верхнего левого угла увеличить на  $a$ , а нижнего правого — уменьшить на  $a$ .

3. Будем использовать цикл **for**, поскольку задано количество квадратов.

IV. Описание переменных:  $a$ ,  $x1$ ,  $y1$ ,  $x2$ ,  $y2$  — integer.

**Пример 17.4\*.** Вывести на экран наибольшее натуральное число из промежутка  $[n, m]$ , которое делится на заданное число  $x$ .

Этапы выполнения задания

I. Исходные данные:  $n$ ,  $m$  (границы промежутка),  $x$  (заданное число).

II. Результат: искомое число или сообщение «Нет таких чисел».

III. Алгоритм решения задачи.



1. Пусть  $i$  — текущее число из промежутка.

2. Поскольку нас интересует наибольшее число из промежутка, то просмотр чисел начнем со значения  $i = m$ . На каждом шаге будем уменьшать  $i$  на 1.

3. Цикл завершится, если мы нашли число, делящееся на  $x$  без остатка (остаток равен нулю), или просмотрели все числа из промежутка  $[n, m]$ .

4. Так как количество повторений заранее неизвестно, используем цикл **while**.

Цикл будет продолжать работу до тех пор, пока условие, сформулированное в пункте 3, будет ложным. А именно: ложным должно быть условие  $(i < n) \text{ or } (i \bmod x = 0)$ . Тогда условие  $\text{not } ((i < n) \text{ or } (i \bmod x = 0))$  будет истинным. Согласно правилам построения отрицаний (см. пример 15.5) это условие можно заменить условием:  $(i \geq n) \text{ and } (i \bmod x \neq 0)$ . Его и будем использовать в качестве условия цикла.

5. Если по окончании цикла  $i = n - 1$ , то нет чисел, удовлетворяющих условию задачи.

IV. Описание переменных:  $n, m, x, i$  — integer.

#### Пример 17.4\*.

V. Программа:

```
var i, n, m, x : integer;
begin
  writeln('Введите границы n, m');
  read(n,m);
  write('Введи x = ');
  read(x);
  i := m;
  while (i >= n) and
    (i mod x <> 0) do
    i := i - 1;
  if i = n - 1 then
    writeln('Нет таких чисел')
  else
    writeln('Искомое число - ',i);
end.
```

VI. Тестирование.

Запустить программу и ввести значения  $n = 10, m = 20, x = 3$ . Результат:

#### Окно вывода

```
Введите границы n, m
10 20
Введи x = 3
Искомое число - 18
```

Запустить программу и ввести значения  $n = 38, m = 45, x = 37$ . Результат:

#### Окно вывода

```
Введите границы n, m
38 45
Введи x = 37
Нет таких чисел
```



1. Что такое оператор цикла?
2. Каким образом можно управлять количеством выполнений тела цикла?
3. Как записывается оператор цикла с предусловием?
4. Как записывается оператор цикла с параметром?



#### Упражнения

1. Измените программу из примера 17.1.

1. Радиусы окружностей равны 20.
2. Окружности располагаются вдоль левого края окна.
3. Радиус окружности вводится пользователем.

4. Окружности образуют рамку вокруг окна.

5\*. Пользователь задает границу окна, вдоль которой будут располагаться окружности (например: 1 — верхняя, 2 — левая, 3 — правая, 4 — нижняя).

2 Какие изменения нужно внести в программу из примера 17.1 для того, чтобы рисунок выглядел следующим образом?



3 Внесите изменения в программу из примера 17.2. Пользователь задает значение второго множителя, а также начальное и конечное значения первого множителя.

4 При каком максимальном значении  $a$  на экране будут видны все 20 квадратов из примера 17.3? Почему при больших значениях  $a$  не видны все квадраты? Измените программу так, чтобы квадраты рисовались от самого маленького к самому большому (установите прозрачную заливку).

5 Какие изменения нужно внести в программу из примера 17.3, чтобы получить следующее изображение? Функции для изменения интенсивности цвета см. в примере 17.1.



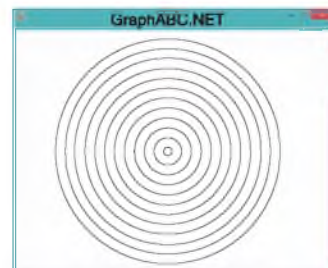
6 Измените программу из примера 17.3. Длина стороны самого большого квадрата 400, а длина стороны каждого следующего квадрата на  $x$  меньше ( $x$  вводится).

7 Напишите программу, которая рисует ряд окружностей заданного радиуса, расположенных по диагонали графического окна. Рассмотрите два варианта:

1. Графическое окно квадратное.

2\*. Графическое окно прямоугольное.

8 Напишите программу, которая рисует концентрические окружности с центром в середине графического окна. Радиус самой маленькой окружности — 10 пикселей. Разница радиусов — 20 пикселей. Используйте изменение интенсивности какого-либо цвета (или двух одновременно) для заливки кругов.



9 В магазине продают конфеты в упаковках по 0.1 кг, 0.2 кг, ... 0.9 кг, 1 кг. Известно, что 1 кг конфет стоит  $x$  рублей. Выведите стоимости каждой упаковки в виде:

0.1 кг конфет стоит ... р.;

0.2 кг конфет стоит ... р. ... .

10\* Выведите на экран наименьшее натуральное число из промежутка  $[n, m]$ , которое является нечетным и не делится на введенное значение  $x$ .

## § 18. Составление алгоритмов для работы с графикой

### 18.1. Расчеты в графических построениях

**Пример 18.1.** Нарисовать прямоугольный треугольник, соответствующий рисунку (катеты треугольника параллельны осям координат). Длины катетов и координаты прямого угла вводятся.



Этапы выполнения задания

I. Исходные данные:  $a$  и  $b$  (длины катетов),  $x$  и  $y$  (координаты вершины прямого угла).

II. Результат: изображение прямоугольного треугольника.

III. Алгоритм решения задачи.

1. Ввод исходных данных.

2. Чтобы изобразить треугольник, нужно выполнить следующие действия:

1) построить линии из точки с координатой  $(x; y)$  в точки с координатами  $(x + a; y)$  и  $(x; y + b)$ ;

2) соединить линией точки  $(x + a; y)$  и  $(x; y + b)$ ;

3) закрасить треугольник. Для закрашивания треугольника нужно знать координаты какой-либо точки внутри треугольника. Такой точкой в

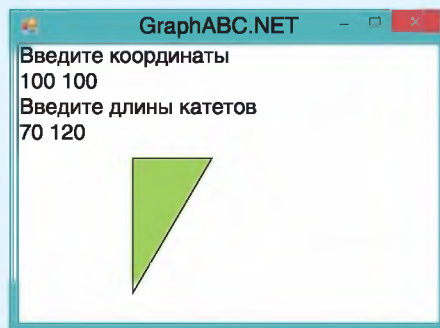
#### Пример 18.1.

V. Программа:

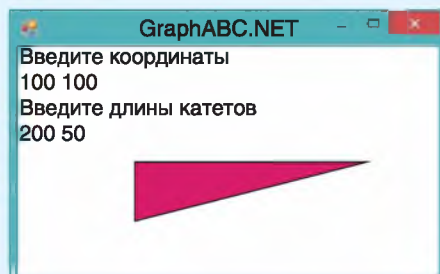
```
uses GraphABC;
var a,b,x,y,x_c, y_c:integer;
begin
  writeln('Введите координаты');
  read(x,y); writeln (x,' ',y);
  writeln('Введите длины катетов');
  read(a,b); writeln (a,' ',b);
  Line(x,y,x+a,y); Line(x,y,x,y+b);
  Line(x+a,y,x,y+b);
  //Координаты точки
  //Внутри треугольника
  x_c := x + 2; y_c := y + 2;
  FloodFill(x_c,y_c,clRandom);
end.
```

VI. Тестирование.

Запустить программу и ввести значения: координаты (100; 100), длины катетов 70 и 120. Результат:



Другой вариант:



**Пример 18.2\*.**

V. Программа:

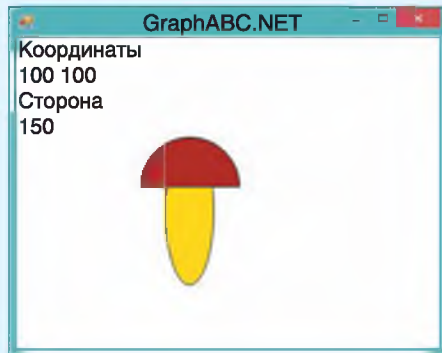
```

uses GraphABC;
var x,y,d: integer;
begin
  writeln('Координаты');
  read(x,y);
  writeln(x,' ',y);
  writeln('Сторона');
  read(d);
  writeln(d);
  SetBrushColor(clYellow);
  Ellipse(x + d div 3,y,
          x + 2 * d div 3,y + d);
  SetBrushColor(clBrown);
  Pie(x + d div 2,y + d div 3,
      d div 3,0,180);
end.

```

VI. Тестирование.

Запустить программу и ввести значения: координаты (100; 100), сторона 150. Результат:



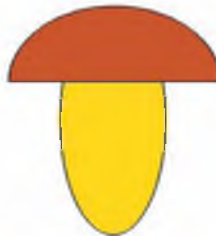
\* Гриб можно вписывать не в квадрат, а в прямоугольник. В этом случае нужно задавать две величины, определяющие размеры прямоугольника: длину ( $d1$ ) и ширину ( $d2$ ).

данном случае может быть точка с координатами  $(x + 2; y + 2)$ <sup>1</sup>.

IV. Описание переменных: все переменные имеют тип `integer`.

Многие графические построения можно обобщить, если предположить, что фигура должна быть вписана в квадрат. В этом случае для построения фигуры достаточно задать координаты  $(x; y)$  верхнего левого угла квадрата и длину его стороны —  $d$ . Используя эти величины, можно получить координаты других вершин квадрата:  $(x + d; y)$ ,  $(x; y + d)$ ,  $(x + d; y + d)$ . Можно получить координаты середины стороны  $(x + d \text{ div } 2; y)$  или центра квадрата  $(x + d \text{ div } 2; y + d \text{ div } 2)$ .

**Пример 18.2\*.** Нарисовать в графическом окне гриб. Задать координаты верхнего левого угла квадрата и длину его стороны для определения местоположения и размеров гриба.



Этапы выполнения задания

I. Исходные данные:  $x$  и  $y$  — координаты верхнего левого угла квадрата, в который вписан гриб,  $d$  — длина его стороны.

II. Результат: изображение гриба.

<sup>1</sup> Для произвольного треугольника можно воспользоваться формулами  $\left( \frac{x_1 + x_2 + x_3}{3}; \frac{y_1 + y_2 + y_3}{3} \right)$ .

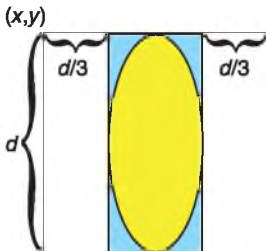


### III. Алгоритм решения задачи.

#### 1. Ввод исходных данных.

2. Для того чтобы построить гриб, нужно выполнить следующие действия:

1) построить овал для изображения ножки гриба. Параметры команды для изображения эллипса определим следующим образом: `ellipse(x + d div 3, y, x + 2*d div 3, y + d);`



2) нарисовать шляпку гриба. Для этого можно использовать команду `Pie` (построение сектора круга). Координаты центра  $(x + \frac{d}{2}; y + \frac{d}{3})$ , радиус —  $\frac{d}{3}$ . Углы равны  $0^\circ$  и  $180^\circ$  соответственно.

IV. Описание переменных: все переменные имеют тип `integer`.

Случайные числа имеют широкое применение в программировании. Они используются в шифровании и в моделировании. Многие компьютерные игры используют случайные числа. На основе случайных чисел генерируются капчи и пароли, реализуются различные лотереи.

В `PascalABC` для получения случайного числа используют функцию `random`. Способы записи функции:

`Random(a, b);` — возвращает случайное целое в диапазоне от  $a$  до  $b$ ;

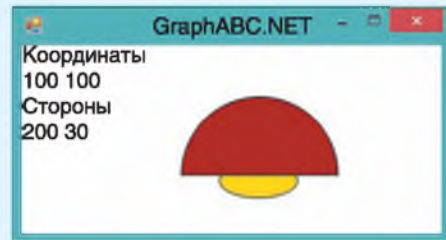
`Random(a);` — возвращает случайное целое в диапазоне от 0 до  $a - 1$ ;

#### Пример 18.2\*. Продолжение.

В программу рисования гриба нужно внести изменения, позволяющие рассчитать положение ножки и шляпки гриба относительно координат  $(x; y)$  и величин  $d1$  и  $d2$ .

```
var x,y,d1,d2: integer;
...
writeln('Стороны');
read(d1, d2);
writeln(d1, ' ', d2);
SetBrushColor(clYellow);
Ellipse(x+d1 div 3, y,
        x + 2 * d1 div 3, y + d2);
SetBrushColor(clBrown);
Pie(x + d1 div 2, y + d2 div 3,
    max(d1,d2) div 3, 0, 180);
```

Результат:



**Случайное число** — число, которое принимает одно значение из множества, причем появление того или иного значения нельзя точно предсказать. Например, если бы числа появились в результате вытягивания бочонков в лото, то такая последовательность чисел была бы случайной.

В языках программирования используют **псевдослучайные числа**, которые получают с использованием генератора случайных чисел — алгоритма, порождающего последовательность чисел, элементы которой почти независимы друг от друга и обычно распределены равномерно на заданном интервале.

**Пример 18.3.**

V. Программа:

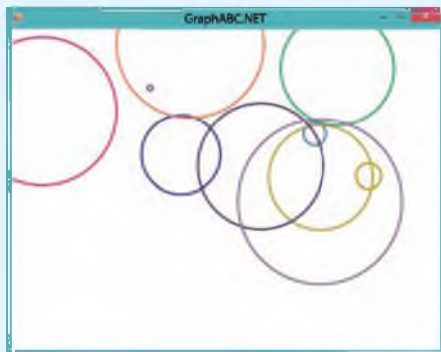
```

uses GraphABC;
var x,y,r: integer;
begin
  SetPenWidth(3);
  SetBrushStyle(bsClear);
  for var i:= 1 to 10 do
  begin
    x := random(600);
    y := random(400);
    R := random(150);
    SetPenColor(clRandom);
    circle(x,y,r);
  end;
end.

```

VI. Тестирование.

Запустить программу. Должно быть нарисовано 10 окружностей. Разные варианты работы программы:



Random; — возвращает случайное вещественное в диапазоне [0..1).

Функция clRandom позволяет задать случайный цвет.

**Пример 18.3.** Написать программу для рисования на экране 10 разноцветных окружностей. Расположение окружностей, их радиусы и цвет определяются случайным образом.

Этапы выполнения задания

I—II. Результатом работы программы, не зависящей от исходных данных, будет изображение 10 окружностей.

III. Алгоритм решения задачи.

1. Установим толщину линий в 3 пикселя и прозрачную заливку.

2. Значения координат центра окружности и ее радиуса определяются функцией random. Значение цвета для границы круга — clRandom.

3. Так как количество повторений известно, будем использовать цикл for.

IV. Описание переменных:  $x$ ,  $y$  (координаты центра),  $r$  (радиус) имеют тип integer.

## 18.2. Использование вспомогательных алгоритмов

Построение фигур можно оформлять в виде вспомогательных алгоритмов. Это позволит использовать такие алгоритмы для решения других задач.

Все графические процедуры, которые использовались ранее, имели параметры. Они позволяли определять местоположение и размер фигур. Пользователь также может составить свой вспомогательный алгоритм с параметрами.

Общий вид процедуры с параметрами:

```
procedure <имя> (<список
                  параметров>:тип);
var ...
begin
  <команды>
end;
```

При вызове процедуры важно помнить, что количество параметров и их порядок должны соответствовать тому, как процедура описана.

**Пример 18.4.** Написать программу для построения  $n$  равнобедренных прямоугольных треугольников с длиной катета  $a$ . Расположение треугольников определяется случайным образом.

Этапы выполнения задания

I. Исходные данные:  $n$  (количество треугольников),  $a$  (длина катета).

II. Результат: изображение  $n$  треугольников.

III. Алгоритм решения задачи.

1. В примере 18.3 изображали окружности, расположенные случайным образом, а в примере 18.1 — прямоугольные треугольники. Воспользуемся программами этих примеров.

2. Ввод значений переменных  $n$  и  $a$ .

3. Так как количество повторений известно, будем использовать цикл **for**.

4. Изменим программу из примера 18.3. Для этого команду **circle** (построение окружности) заменим на команду построения треугольника:

1) местоположение треугольника задается координатами прямого угла, которые определим случайным образом;

2) катеты прямоугольного треугольника имеют одинаковую длину — значение  $a$ .

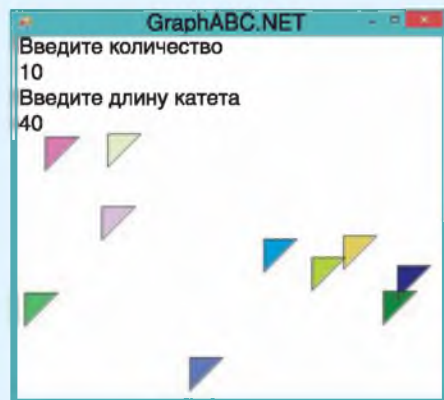
#### Пример 18.4.

V. Программа:

```
uses GraphABC;
var n, x, y, a : integer;
procedure pr_treug (x, y, a,
                    b : integer);
var x_c, y_c:integer;
begin
  line(x, y, x + a,y);
  line(x, y, x, y + b);
  line(x + a, y, x, y + b);
  x_c := x + 2; y_c := y + 2;
  FloodFill(x_c,y_c,clRandom);
end;
begin
  writeln('Введите количество');
  read(n); writeln (n);
  writeln('Введите длину катета');
  read(a); writeln (a);
  for var i:= 1 to n do
    begin
      x:= random(500);
      y:= random(400);
      pr_treug(x, y, a, a);
    end;
  end.
```

VI. Тестирование.

Запустить программу. Результат:



При вводе других значений результаты будут иными.

**Пример 18.5\*.**

V. Программа:

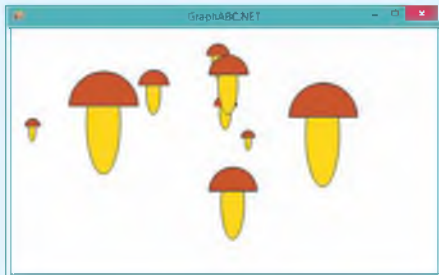
```

uses GraphABC;
var x,y,d: integer;

procedure grib(x,y,d:integer);
begin
  SetBrushColor(clYellow);
  Ellipse(x + d div 3, y,
    x + 2*d div 3, y + d);
  SetBrushColor(clBrown);
  Pie(x + d div 2, y + d div 3,
    d div 3, 0, 180);
end;

Begin
  for var i: = 1 to 10 do
    begin
      x:= random(400);
      y:= random(200);
      d:= random(150);
      grib(x,y,d);
    end;
  end.
  VI. Тестирование.
  Результат может быть следующим:

```



Можно добавить раскраску случайным цветом:



5. Построение одного прямоугольного треугольника опишем во вспомогательном алгоритме `pr_treug`. Параметры процедуры построения треугольника — координаты вершины прямого угла и длины катетов. Алгоритм описан в примере 18.2.

IV. Описание переменных: все переменные имеют тип `integer`.

**Пример 18.5\*.** Нарисовать 10 грибов. Расположение и их размеры определяются случайным образом.

Этапы выполнения задания

I—II. Результатом работы программы, не зависящим от исходных данных, будет изображение 10 грибов.

III. Алгоритм решения задачи.

1. Построение одного гриба опишем во вспомогательном алгоритме. Алгоритм описан в примере 18.2.

2. Значения координат верхнего левого угла и размер гриба определяются функцией `random`.

3. Так как количество повторений известно, будем использовать цикл `for`.

IV. Описание переменных:  $x$ ,  $y$  (координаты верхнего левого угла),  $d$  (размер) — `integer`.

**Пример 18.6.** Заполнить графическое окно окружностями с радиусом 10.

Этапы выполнения задания

I—II. Исходные данные отсутствуют. Окружности должны заполнить все графическое окно.

III. Алгоритм решения задачи.

1. Задача является обобщением задачи примера 17.1. Команды программы следует повторить для нескольких рядов окружностей. Количество рядов определяется высотой окна. Рисование



одного ряда оформим как вспомогательный алгоритм `row`.

2. Положение любого ряда окружностей определяется координатой  $y$ . Для каждого значения  $y$ , пока он не станет большим, чем вертикальный размер экрана, выполняем в цикле следующее:

- 1) рисуем ряд окружностей;
- 2) изменяем  $y$ .

IV. Описание переменных:  $x, y, r$  — integer.

В примере 18.6 показано, как заполнить графическое окно окружностями. Внеся небольшие изменения в эту программу, можно заполнять графическое окно любыми другими фигурками. Для этого достаточно заменить команду `Circle(x,y,r)` в процедуре `row` на другую команду. Можно выбрать графический примитив из библиотеки `GraphABC` или самостоятельно написать процедуру рисования фигурки (например, использовать процедуру рисования гриба из примера 18.5).

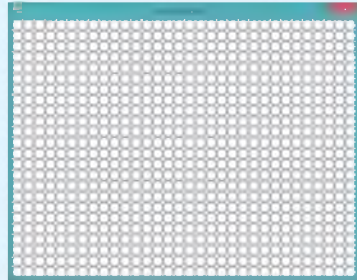
#### Пример 18.6.

V. Программа:

```
uses GraphABC;
var x, y, r : integer;
procedure row(y : integer);
begin
  x := 10; R := 10;
  while x < WindowWidth do
  begin
    Circle(x,y,r); x := x+20;
  end;
end;
begin
  y := 10;
  while y <= WindowHeight do
  begin
    Row(y); y := y + 20;
  end;
end.
```

VI. Тестирование.

Результат:



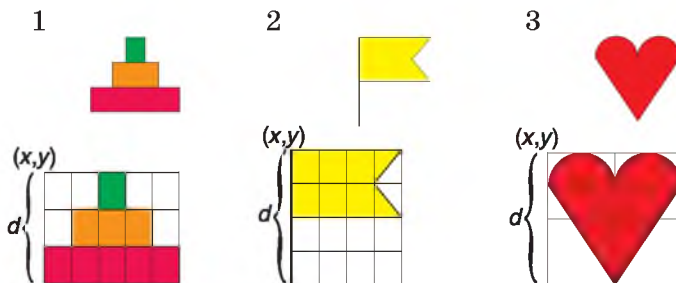
1. Как задать случайное число?
2. Как задать случайный цвет?
3. Как описать процедуру с параметрами?



#### Упражнения

- 1 Выполните задания для примера 18.1.
  1. Поэкспериментируйте с программой, вводя различные значения исходных данных.
  2. Объясните, что происходит при вводе отрицательных значений длин катетов.
  3. Что произойдет, если ввести отрицательные значения координат? Объясните результат.
- 2 Выполните задания для примера 18.3.
  1. Выполните программу несколько раз. Уберите прозрачную заливку. Объясните, почему некоторые окружности не видны.

2. Внесите в программу такие изменения, чтобы можно было изобразить 20 кругов; 100 кругов.
3. Какой максимальный размер может иметь радиус круга в программе? Внесите в программу изменения так, чтобы рисовались круги с радиусом не более 20. Количество кругов установите равным 10 000.
4. Внесите изменения в программу так, чтобы пользователь мог вводить количество отображаемых на экране кругов.
- 3 Выполните задания для примера 18.4.
  1. Запустите программу несколько раз. Объясните, почему при некоторых запусках треугольники рисуются поверх текста в верхнем левом углу экрана. Измените программу так, чтобы треугольники рисовались ниже текста (правее текста).
  2. Добавьте в программу возможность ввода длины второго катета.
  3. Измените программу так, чтобы длины катетов задавались случайным образом.
- 4 Напишите программу, которая строит случайным образом изображения 20 горизонтальных отрезков длиной 30 пикселей. Разработайте два варианта решения задачи. Один с использованием цикла `while`, а другой — цикла `for`.
  1. Сравните две программы решения задачи. Какой вариант решения данной задачи представляется вам лучшим? Почему?
  2. Задайте в программе толщину отрезка в 3 пикселя.
  3. Какие изменения нужны в программе, чтобы толщина отрезка была случайным числом из промежутка  $[2; 8]$ ?
  4. Внесите изменения в программу так, чтобы пользователь мог вводить количество отображаемых на экране отрезков.
  5. Какие изменения нужно внести в программу, чтобы вместо горизонтальных отрезков изображались вертикальные? Диагональные?
- 5 Используя процедуру рисования треугольника из примера 18.4, нарисуйте ряд треугольников вдоль верхнего (левого) края графического окна.
- 6\* Напишите программу для рисования одной из фигурок. Задаются координаты верхнего левого угла и длина стороны квадрата (длины сторон прямоугольника):



**7\*** Напишите программу, которая нарисует ряд фигурок вдоль края графического окна. Используйте фигурки, которые рисовали в задании 6.

**8** Выполните задания для примера 18.6.

1. Измените в программе значение  $r = 10$  на  $r = 20$ . Почему получился такой рисунок? Поэкспериментируйте со значениями радиуса, установив прозрачную заливку.

2. Какие изменения нужно внести в программу, чтобы экран заполнялся кругами с радиусом 20 без пересечений?

3. Внесите изменения в программу так, чтобы все круги были красными или разноцветными.

4. Внесите в программу изменения так, чтобы графическое окно можно было заполнять кругами введенного радиуса.

**9\*** Напишите программу, которая заполнит все графическое окно:

1. Грибами (пример 18.2).

2. Фигурками из задания 6.

## § 19. Использование основных алгоритмических конструкций для решения практических задач

### 19.1. Использование числовых последовательностей

Числовые последовательности позволяют описывать многие процессы, происходящие в природе и обществе.

Например, последовательность чисел 2, -1, 0, 2, 0, 1, -2 может задавать значения температуры по дням недели; последовательность 746, 751, 758 — значения средней заработной платы сотрудников предприятия за квартал и т. д.

Последовательности могут задаваться формулой, в которой значение элемента зависит от того, какой у него номер в последовательности (пример 19.1).

Другим способом задания элементов последовательности является определение значения нового элемента

Подтверждением важности числовых последовательностей является тот факт, что создана целая энциклопедия числовых последовательностей<sup>1</sup>.

**Пример 19.1.** Элементы последовательности нечетных положительных чисел можно описать с помощью формулы  $a_n = 2n - 1$ . В этой формуле  $n$  — номер элемента в последовательности. Минимальное значение числа  $n = 1$ . Используя формулу, получим последовательность: 1, 3, 5, 7, ... .

Элементы последовательности могут быть действительными числами. Например, формула  $a_n = \frac{n}{n^2 + 1}$  задает следующую последовательность: 0.5, 0.4, 0.3, 0.235, 0.192, ... .

<sup>1</sup> Онлайн-энциклопедия целочисленных последовательностей. Режим доступа: <http://oeis.org/?language=russian> (дата доступа: 17.01.2018).

**Пример 19.2.** Одной из наиболее известных последовательностей является последовательность Фибоначчи: 1, 1, 2, 3, 5, 8, 13, ... . Несложно заметить, что каждый ее элемент, начиная с третьего, равен сумме двух предыдущих. Это можно записать так:  $f_n = f_{n-1} + f_{n-2}$ ,  $f_1 = 1$ ,  $f_2 = 1$ .

**Пример 19.3.** Рассмотрим последовательность 2, 4, 8, 16, ... . Каждое число в этой последовательности является степенью числа 2, поэтому можно задать последовательность формулой  $a_n = 2^n$ . С другой стороны, каждый элемент последовательности, начиная со второго, в два раза больше предыдущего. Получим формулу  $a_n = 2a_{n-1}$  (для  $n > 1$ ,  $a_1 = 2$ ).

**Пример 19.4.**

V. Программа:

```
var k, a: integer;
begin
  write('Количество k = '); read(k);
  for var n := 1 to k do
    begin
      a := 2*n; write(a, ' ');
    end;
  end.
```

VI. Тестирование.

Запустить программу и ввести значения  $k = 5$ . Результат:

Окно вывода

```
Количество k = 5
2 4 6 8 10
```

Запустить программу и ввести значение  $k = 100$ . Результат:

Окно вывода

```
Количество k = 100
2 4 6 8 10 12 14 16 18 20 22 24 26 28
30 32 34 36 38 40 42 44 46 48 50 52
54 56 58 60 62 64 66 68 70 72 74 76
78 80 82 84 86 88 90 92 94 96 98 100
102 104 106 108 110 112 114 116 118
120 122 124 126 128 130 132 134 136
138 140 142 144 146 148 150 152 154
156 158 160 162 164 166 168 170 172
174 176 178 180 182 184 186 188 190
192 194 196 198 200
```

через значение предыдущего (пример 19.2).

Есть последовательности, которые можно задавать как первым, так и вторым способом (пример 19.3). Последовательности могут строиться из случайных чисел.

**Пример 19.4.** Вывести на экран первые  $k$  четных чисел.

Этапы выполнения задания

I. Исходные данные:  $k$  (количество чисел).

II. Результат:  $k$  четных чисел, начиная с 2.

III. Алгоритм решения задачи.

1. Ввод числа  $k$ .

2. Для получения четного числа запишем формулу  $a_n = 2n$ .

3. Так как количество чисел заранее известно, то для их получения можно воспользоваться циклом **for**.

4. Текущее число будем хранить в переменной  $a$ . Значение  $a$  вычисляется по формуле и зависит от значения  $n$  — счетчика цикла. Переменная  $n$  будет изменяться от 1 (номер первого четного числа) до  $k$  (номер последнего числа).

5. Полученные числа будем выводить в цикле через пробел.

IV. Описание переменных:  $k$ ,  $a$  — integer.

**Пример 19.5.** Вывести на экран все элементы последовательности Фибоначчи меньше  $x$  ( $x$  вводится).

Этапы выполнения задания

I. Исходные данные:  $x$  (граница для чисел).

II. Результат: числа Фибоначчи меньше  $x$ .



## III. Алгоритм решения задачи.

1. Ввод числа  $x$ .

2. Вывод первых двух элементов.

3. Числа Фибоначчи, начиная с третьего, получают по формуле  $a_n = a_{n-1} + a_{n-2}$  ( $a_1 = a_2 = 1$ ). Для вывода чисел понадобятся три переменные: значение  $a$ , которое нужно вывести, и два предыдущих значения —  $b$  и  $c$ .

$n$	1	2	3	4	5	6	7
Числа Фибоначчи	1	1	2	3	5	8	13
Текущий шаг			$c$	$b$	$a$		
Следующий шаг				$c$	$b$	$a$	

После вывода значения  $a$  нужно «сдвинуть» значения переменных:  $c := b$ ;  $b := a$ . Начальные значения:  $c := 1$ ;  $b := 1$ ;  $a := 2$ .

4. Так как количество чисел заранее неизвестно, то для их вычисления нужно использовать цикл **while**. Условие продолжения работы цикла:  $a < x$ .

5. В цикле выводим текущее значение  $a$ , «сдвигаем» значения переменных и получаем новое значение  $a$ .

IV. Описание переменных:  $x, a, b, c$  — integer.

Числа Фибоначчи обладают множеством интересных математических свойств. Например:

1. Могут быть вычислены по формуле

$$f_n = \frac{\left(\frac{1+\sqrt{5}}{2}\right)^n - \left(\frac{1-\sqrt{5}}{2}\right)^n}{\sqrt{5}}.$$

2. Отношение следующего числа к предыдущему является постоянной величиной  $\approx 1.618$  (начиная с 13-го). Это число называют золотым сечением.

3. Для трех последовательных чисел Фибоначчи верно соотношение Кассини:  $f_{n+1}f_{n-1} - f_n^2 = (-1)^n$ .

## Пример 19.5.

V. Программа:

```
var a, b, c, x: integer;
begin
  write('Граница x = '); read(x);
  c := 1; b := 1; a := 2;
  write(c, ' ', b, ' ');
  while a < x do
  begin
    write(a, ' ');
    c := b; b := a; a := b + c;
  end;
end.
```

VI. Тестирование.

Запустить программу и ввести значение  $x = 100$ . Результат:

## Окно вывода

Граница x = 100

1 1 2 3 5 8 13 21 34 55 89

Чтобы узнать, сколько чисел получили в качестве результата, определим переменную  $k$ . Переменная будет увеличивать свое значение на 1 каждый раз, когда выводится очередное число. После завершения цикла можно вывести значение  $k$ .

```
var a, b, c, x, k: integer;
begin
  write('Граница x = ');
  read(x);
  c := 1; b := 1; a := 2;
  write(c, ' ', b, ' ');
  k := 2;
  while a < x do
  begin
    write(a, ' '); k := k+1;
    c := b; b := a; a := b + c;
  end;
  writeln;
  write('k = ', k);
end.
```

Результат для  $x = 1000$ .

## Окно вывода

Граница x = 1000

1 1 2 3 5 8 13 21 34 55 89 144 233

377 610 987

k = 16

**Пример 19.6.**

V. Программа:

```

var n, k, a: integer;
begin
  write('Количество билетов n = ');
  read(n); k := 0;
  for var i:= 1 to n do
    begin
      a:= random(1,100); write(a, ' ');
      if a mod 5 = 0 then
        k := k+1;
    end;
    writeln;
    writeln('Выиграло ', k,
      ' билета(-ов)');
  end.

```

VI. Тестирование. Запустить программу и ввести значение  $n = 20$ .  
Результат:

**Окно вывода**

```

Количество билетов n = 20
56 81 10 34 3 20 43 34 57 54 92 59 47
23 44 90 83 79 29 91
Выиграло 3 билета(-ов)

```

При одном и том же значении  $n$  программа может выдавать различные результаты, поскольку числа получаются случайным образом.

**Пример 19.7.**

V. Программа:

```

var m : integer;
    a, S: real;
begin
  write('Количество дней m = ');
  read(m); S := 0;
  for var n := 1 to m do
    begin
      a := n*n*n/(sqrt(n*n*n)-n+1);
      S := S+a;
    end;
    writeln('Всего бактерий =
      = ', S, ' млн');
  end.

```

VI. Тестирование.

Запустить программу и ввести значение  $m = 3$ . Результат:

**Окно вывода**

```

Количество дней m = 3
Всего бактерий = 13.8230024772972 млн

```

**Пример 19.6.** Катя и Петя решили организовать благотворительную лотерею. Для этого они случайным образом генерируют номер билета. Номера билетов принадлежат промежутку  $[1..100]$ . Выигрышным билетом будет тот, номер которого кратен 5. Определить, сколько будет выигрышных билетов среди  $n$  сгенерированных Катей и Петей.

Этапы выполнения задания

I. Исходные данные:  $n$  (количество билетов).

II. Результат:  $k$  — количество выигрышных билетов.

III. Алгоритм решения задачи.

1. Ввод числа  $n$ .

2. До начала генерации количество выигрышных билетов равно нулю ( $k := 0$ ).

3. Так как количество билетов заранее известно, то для получения их номеров можно воспользоваться циклом **for**.

4. Текущий номер билета будем хранить в переменной  $a$ . Номера билетов будем получать по формуле  $a = \text{Random}(1,100)$  и выводить на экран. Для каждого номера будем проверять, равен ли 0 остаток от деления числа на 5. И если равен, то увеличим на 1 значение переменной  $k$ .

5. Вывод результата.

IV. Описание переменных:  $n, k, a$  — integer.

## 19.2. Нахождение суммы элементов числовой последовательности

**Пример 19.7.** В лаборатории выводят полезные бактерии. Экспериментально

было установлено, что количество бактерий (в млн) зависит от номера дня, в который проводится эксперимент, следующим образом:  $a_n = \frac{n^3}{\sqrt{n^3 - n + 1}}$ .

Определите, сколько бактерий вывели за  $m$  дней.

Этапы выполнения задания

I. Исходные данные:  $m$  (число дней).

II. Результат:  $S$  (общее количество бактерий).

III. Алгоритм решения задачи.

1. Ввод числа  $m$ .

2. Для вычисления общего количества бактерий необходимо последовательно прибавлять количество бактерий, выведенных в текущий день, к уже полученному количеству бактерий. Начальное значение суммы равно 0.

3. Так как количество бактерий заранее известно, для вычисления суммы можно воспользоваться циклом **for**.

4. Количество бактерий в текущий день будем хранить в переменной  $a$ . Значение  $a$  зависит от значения  $n$  — счетчика дней. Переменная  $n$  изменяется от 1 до  $m$ .

5. Вывод результата  $S$ .

IV. Описание переменных:  $m$  — integer,  $S$ ,  $a$  — real.

### 19.3. Возведение числа в степень

**Пример 19.8.** Возвести вещественное число  $a$  в целую степень  $n$ .

Этапы выполнения задания

I. Исходные данные:  $a$  (основание степени),  $n$  (показатель степени).

II. Результат:  $S$  (значение степени).

III. Алгоритм решения задачи.

#### Пример 19.7. Продолжение.

Для проверки правильности результата можно посчитать значение суммы на калькуляторе:

$$a_1 = 1; a_2 = \frac{2^3}{\sqrt{8 - 2 + 1}} \approx 4.38;$$

$$a_3 = \frac{3^3}{\sqrt{27 - 3 + 1}} \approx 8.44; S \approx 13.82.$$

Запустить программу и ввести значение  $m = 30$ . Результат:

#### Окно вывода

Количество дней  $m = 30$

Всего бактерий = 2613.36198051392 млн

#### Пример 19.8.

V. Программа:

```
var n, m: integer;
    a, S: real;
begin
    write('Основание a = '); read(a);
    write('Показатель n = '); read(n);
    S := 1; m := abs(n);
    for var i:= 1 to m do
        S := S*a;
    if n<0 then S:= 1/S;
    writeln('Степень = ',S);
end.
```

VI. Тестирование программы.

Запустить программу и ввести значения  $a = 5$ ,  $n = 3$ . Результат:

#### Окно вывода

Основание  $a = 5$

Показатель  $n = 3$

Степень = 125

Запустить программу и ввести значения  $a = 3$ ,  $n = 0$ . Результат:

#### Окно вывода

Основание  $a = 3$

Показатель  $n = 0$

Степень = 1

Запустить программу и ввести значения  $a = 5$ ,  $n = -2$ . Результат:

#### Окно вывода

Основание  $a = 5$

Показатель  $n = -2$

Степень = 0.04

VII. Правильность вычислений проверить на калькуляторе.

**Пример 19.9.**

V. Программа:

```

var k:integer;
    x,y,h:real;
begin
  writeln('Количество значений');
  readln(k); x := -3; h := 0.5;
  for var n:= 1 to k do
  begin
    y := (x+2)/(x*x+3);
    writeln(x:7:2,y:10:3); x := x+h;
  end;
end.

```

VI. Тестирование программы.

Запустить программу и ввести значение  $k = 5$ . Результат:

**Окно вывода****Количество значений**

5

-3.00	-0.083
-2.50	-0.054
-2.00	-0.000
-1.50	-0.059
-1.00	-0.250

Добавим в программу вывод границ таблицы:

```

var k:integer;
    x,y,h: real;
begin
  writeln('Количество значений');
  readln(k); x := -3; h := 0.5;
  writeln('-----');
  writeln('| x | y |');
  writeln('-----');
  for var n := 1 to k do
  begin
    y := (x+2)/(x*x+3);
    writeln('|',x:7:2,'|',y:10:3,'|');
    x := x+h;
  end;
  writeln('-----');
end.

```

Результат при  $k = 4$ :**Окно вывода****Количество значений**

4

x	y
-3.00	-0.083
-2.50	-0.054
-2.00	0.000
-1.50	0.095

1. Ввод чисел  $a, n$ .

2. Для возведения числа в целую неотрицательную степень нужно последовательно умножать на основание степени то значение, которое получили на предыдущем шаге. Это вытекает из равенства:  $a^n = a^{n-1} \cdot a$ .

3. Если степень отрицательная, то результат можно получить из равенства:  $a^{-n} = \frac{1}{a^n}$ . Количество повторений цикла  $m$  можно определить как  $m = |n|$ .

4. Для вычисления произведения можно воспользоваться циклом **for**. Начальное значение степени равно 1.

5. В конце проверим, является ли значение  $n$  положительным или отрицательным. Если оно отрицательно, то нужно изменить значение переменной  $S$ .

6. Вывод результата  $S$ .

IV. Описание переменных:  $m, n$  — integer,  $a, S$  — real.

### 19.4. Построение таблицы значений функции

**Пример 19.9.** Вывести на экран таблицу значений функции  $y = \frac{x+2}{x^2+3}$ .

Количество значений вводится. Начальное значение  $x = -3$ , значения аргумента выводятся с шагом  $h = 0,5$ .

Этапы выполнения задания

I. Исходные данные:  $k$  (количество точек).

II. Результат:  $k$  значений аргумента и соответствующих им значений функции.

III. Алгоритм решения задачи.

1. Ввод числа  $k$ .

2. Для получения таблицы нужно в цикле вычислять и выводить значение



аргумента и соответствующее ему значение функции:

1) начальное значение аргумента  $x = -3$ . Для получения очередного значения аргумента нужно к текущему значению прибавить шаг  $h$ ;

2) значение функции вычисляется по формуле  $y = \frac{x+2}{x^2+3}$ ;

3) полученные значения выводятся на экран. Для того чтобы значения выводились строго одно под другим, нужно использовать форматный вывод. Для этого задать количество позиций для вывода значения и определить количество цифр после запятой. Запись  $x:7:2$  означает, что для вывода переменной используется 7 позиций, после запятой выводятся 2 цифры.

3. Поскольку количество точек известно, воспользуемся циклом **for**.

IV. Описание переменных:  $k$  — integer,  $x, y, h$  — real.

### 19.5. Выделение цифр из числа

**Пример 19.10.** Дано натуральное число  $n$ . Вывести цифры числа по одной в строке (начиная с разряда единиц). Определить, сколько цифр в числе.

Этапы выполнения задания

I. Исходные данные:  $n$  (число).

II. Результат:  $z$  (текущая цифра числа),  $k$  (количество цифр в числе  $n$ ).

III. Алгоритм решения задачи.

1. Ввод исходных данных — число  $n$ .

2. Определение начального значения счетчика для количества цифр ( $k := 0$ ).

3. Количество цифр числа равно количеству десятичных разрядов в числе.

Понятие числа возникло в глубокой древности из практической потребности людей. Для записи чисел используют цифры. Любая информация в компьютере представляется с помощью всего двух цифр (0 и 1).

Числовой код имеет каждая страна мира, цифры задают пин-код банковской карты. Сегодня с помощью цифр можно получить числовой образ практически любого объекта.

#### Пример 19.10.

V. Программа:

```
var k,n,z: integer;
begin
  write('Введите n = ');
  read(n);
  k := 0;
  while n > 0 do
  begin
    //Текущая цифра
    z := n mod 10;
    writeln(z);
    //Уменьшение числа в 10 раз
    n := n div 10;
    //Подсчет кол-ва цифр
    k := k + 1;
  end;
  writeln('в числе ', k,
    ' цифр (-a/-ы)');
end.
```

VI. Тестирование.

Запустить программу и ввести значение  $n = 13579$ . Результат:

```
Окно вывода
Введите n = 13579
9
7
5
3
1
в числе 5 цифр (-a/-ы)
```

Запустить программу и ввести значение  $n = 1$ . Результат:

```
Окно вывода
Введите n = 10
1
в числе 1 цифр (-a/-ы)
```

Алгоритм Евклида — алгоритм для нахождения наибольшего общего делителя двух целых чисел. Алгоритм назван в честь древнегреческого математика Евклида (III в. до н. э.), который впервые описал его в книгах «Начала». Это один из старейших численных алгоритмов, используемых в наше время.

Евклид использовал данный алгоритм не только для чисел, но и для геометрических величин: длин, площадей, объемов. В XIX в. алгоритм был обобщен на другие математические объекты. Сегодня алгоритм Евклида используется при шифровании данных.

**Пример 19.11.** Алгоритм Евклида для чисел 42 и 24:

№	$a$	$b$
1	42	24
2	18 (42–24)	24
3	18	6 (24–18)
4	12 (18–6)	6
5	6 (12–6)	6

**Пример 19.12.**

V. Программа:

```
var a,b:integer;
begin
  write('Значки у Иры a = ');
  read(a);
  write('Значки у Игоря b = ');
  read(b);
  while a<>b do
    if a>b then
      a := a - b
    else
      b := b - a;
  writeln('Всего друзей = ',a);
end.
```

VI. Тестирование.

Запустить программу и ввести значения:  $a = 42$ ,  $b = 24$ . Результат:

Окно вывода

```
Значки у Иры a = 42
Значки у Игоря b = 24
Всего друзей = 6
```

Для нахождения каждой цифры числа нужно:

- разделить число на 10;
- найти целую часть от деления и остаток (остаток и будет очередной цифрой) и увеличить счетчик количества цифр;
- вывести полученную цифру;
- поскольку количество цифр в числе заранее неизвестно, будем использовать цикл **while**. Пока целая часть от деления больше 0, в числе еще есть цифры и нужно перейти к выполнению пункта а), иначе все цифры найдены.

4. Вывод значения переменной  $k$ .

IV. Описание переменных:  $k$ ,  $n$ ,  $z$  — integer.

## 19.6. Наибольший общий делитель двух чисел

Наибольшим общим делителем (НОД) для двух целых чисел называют наибольший из их общих делителей. Пример: для чисел 42 и 24 наибольший общий делитель равен 6.

Существуют несколько алгоритмов нахождения НОД. С одним из них вы познакомились на уроках математики. Нужно разложить каждое из чисел на простые множители, выбрать общие и перемножить.

Рассмотрим другой алгоритм, который называется алгоритм Евклида.

1. Из большего числа вычитаем меньшее.

2. Если получается 0, то числа равны друг другу и это значение является НОД.

3. Если результат вычитания не равен 0, то большее число заменяем на разность большего и меньшего.

4. Переходим к пункту 1.

(Рассмотрите пример 19.11.)

**Пример 19.12.** Ира и Игорь коллекционируют значки. У Иры в коллекции  $a$  значков, а у Игоря —  $b$ . Поскольку значков много, ребята решили поделиться своими значками с друзьями. Какое наибольшее количество общих друзей может быть у Иры и Игоря, если каждый из них хочет разделить все свои значки между друзьями без остатка? Например, если  $a = 42$  и  $b = 24$ , то значки можно разделить, если у Иры и Игоря 1, 2, 3 или 6 общих друзей. Наибольшее количество — 6.

Этапы выполнения задания

I. Исходные данные:  $a$  и  $b$  (количество значков у Иры и у Игоря).

II. Результат: наибольшее количество общих друзей.

III. Алгоритм решения задачи.

1. Ввод чисел  $a, b$ .

2. Поскольку значки нужно делить без остатка, то ответом на задачу может быть только общий делитель чисел  $a$  и  $b$ . Среди всех делителей нужно найти наибольший.

3. Для решения задачи напомним программу вычисления НОД( $a, b$ ) по алгоритму Евклида. Пока числа  $a$  и  $b$  не равны, выполняем следующее:

1) сравниваем два числа;

2) если  $a > b$ , заменяем  $a$  на разность  $a - b$ , иначе заменяем  $b$  на разность  $b - a$ .

4. Вывод результата. Вывести можно как значение  $a$ , так и  $b$  (поскольку они равны).

IV. Описание переменных:  $a, b$  — integer.

**Пример 19.13\*.** Написать программу вычисления НОД( $x, y, z$ ).

**Пример 19.12. Продолжение.**

Для значений  $a = 1449$ ,  $b = 596$  получим:

#### Окно вывода

```
Значки у Иры a = 1449
Значки у Игоря b = 596
Всего друзей = 1
```

Данный результат означает, что числа 1449 и 596 взаимно простые и все значки можно отдать только одному другу.

\*В Pascal, кроме процедур, часто используются вспомогательные алгоритмы в виде функций. Функция всегда возвращает значение, которое нужно присвоить какой-либо переменной или использовать в любом выражении:  
 $b := \text{abs}(x)$ ,  $t := 2 + \text{sqrt}(a)$ .

Общий вид функции:

```
function <имя>(<список
параметров>:тип): тип результата;
var ... //Может отсутствовать
begin
    <команды>
    <имя> := <значение>;
end;
```

**Пример 19.13\*.**

V. Программа:

```
var x,y,z,d,f:integer;
function NOD
    (a,b:integer):integer;
begin
    while a<>b do
        if a>b then
            a := a-b
        else
            b := b-a;
    NOD := a;
end;
begin
    write('Введите x = ');
    read(x);
    write('Введите y = ');
    read(y);
    write('Введите z = ');
    read(z);
    d := NOD(x,y);
    f := NOD(d,z);
    writeln('НОД = ',f);
end.
```

**Пример 19.13\*. Продолжение.****VI. Тестирование.**

Запустить программу и ввести значения:  $x = 26$ ,  $y = 143$ ,  $z = 65$ . Результат:

Окно вывода
Введите $x = 26$
Введите $y = 143$
Введите $z = 65$
НОД = 13

Запустить программу и ввести значения:  $x = 354$ ,  $y = 847$ ,  $z = 125$ . Результат:

Окно вывода
Введите $x = 354$
Введите $y = 847$
Введите $z = 125$
НОД = 1

Данный результат означает, что числа 354, 847 и 125 взаимно простые.

**Этапы выполнения задания**

I. Исходные данные:  $x$ ,  $y$  и  $z$  (три числа).

II. Результат: НОД ( $x$ ,  $y$ ,  $z$ ).

III. Алгоритм решения задачи.

1. Ввод чисел  $x$ ,  $y$ ,  $z$ .

2. Используем то, что  $\text{НОД}(x, y, z) = \text{НОД}(\text{НОД}(x, y), z)$ . То есть сначала вычислим  $d = \text{НОД}(x, y)$ , а затем  $f = \text{НОД}(d, z)$ .

3. Для вычисления НОД двух чисел составим функцию  $\text{NOD}(a, b)$ . Команды функции  $\text{NOD}$  рассмотрены в примере 19.12.

4. Вывод результата.

IV. Описание переменных:  $x$ ,  $y$ ,  $z$ ,  $d$ ,  $f$  — integer.

**Упражнения**

1 Выполните задания для примера 19.4.

- Внесите в программу изменения так, чтобы числа выводились в обратном порядке — от большего к 2.
- Какие изменения нужно внести в программу, чтобы вывести все четные числа меньше введенного числа  $x$ ?
- Внесите изменения в программу так, чтобы выводились числа, кратные 3, 5; кратные введенному числу  $x$ .
- Измените программу так, чтобы можно было вывести все числа последовательности, заданной формулой  $a_n = \frac{n}{n^2 + 1}$  (пример 19.1).

2 Выполните задания для примера 19.5.

- Запустите программу для разных значений  $x$ .
- Какое максимальное значение  $x$  можно ввести?
- Замените в программе тип `integer` на тип `int64`. Сколько чисел последовательности Фибоначчи можно найти теперь?
- Измените программу так, чтобы она выводила значение числа Фибоначчи по введенному номеру.

3 Выполните задания для примера 19.6.

- Запустите программу несколько раз для одного и того же значения (например, 20). Какое число получается в ответе чаще всего? Почему?



2. Измените диапазон случайных чисел на  $[1, 1000]$  и проверьте, какое число получается в результате чаще других для тех же 20 элементов.
3. Измените программу так, чтобы вычислялось количество чисел, кратных введенному числу  $x$ .
4. Измените программу так, чтобы можно было посчитать количество чисел из промежутка  $[a; b]$ ,  $a$  и  $b$  вводятся.
- 4\* Напишите программу, которая будет выводить на экран элементы последовательности трибоначчи — первые элементы последовательности: 0, 0, 1, 1, 2, 4, 7, 13, 24, 44, 81, 149, ... . Каждый элемент, начиная с четвертого, равен сумме трех предыдущих:  $a_n = a_{n-1} + a_{n-2} + a_{n-3}$ .
  1. По заданному  $n$  вывести элемент последовательности.
  2. Для заданного  $x$  вывести элементы последовательности меньше  $x$ .
- 5 Выполните задания для примера 19.7.
  1. Замените в решении задачи цикл **for** на цикл **while**.
  2. Выполните программу для  $m = 2000$ . Почему в ответе сумма = NaN? Какие изменения нужно внести в программу для получения результата? Найдите значение для  $m = 2000$ ,  $m = 10\,000$ .
- 6 Найдите сумму первых  $m$  элементов последовательности. Число  $m$  вводится. Элементы последовательности задаются формулой  $a_n = \frac{1}{n^3}$ .
  1. На сколько большим должно быть значение  $m$ , чтобы программа выдала ответ «сумма = Infinity»? Почему так произошло?
  2. Какие изменения нужно внести в программу для получения правильного результата?
  3. Замените в решении задачи цикл **for** на цикл **while**. Найдите сумму для  $m = 20\,000$ ,  $m = 1\,000\,000$ .
- 7 Выполните задание для примера 19.8. Измените форму вывода результата таким образом, чтобы результат выводился в виде  $a^n = S$  (например, для значений  $a = 2$ ,  $n = 3$  должно быть напечатано  $2^3 = 8$ ).
- 8 Факториалом числа  $n$  называют произведение всех натуральных чисел, не превосходящих  $n$ . Обозначают факториал так:  $n!$ . По определению факториал числа 0 равен 1. Напишите программу, которая вычислит значение факториала целого неотрицательного числа  $n$ . Для проверки можно использовать следующее:  $0! = 1$ ;  $2! = 2$ ;  $5! = 120$ ;  $10! = 3\,628\,800$ .
- 9 Выполните задание для примера 19.9. Замените в решении задачи цикл **for** на цикл **while**. В качестве условия в цикле **while** можно использовать следующее:  $x <= 3$ .
- 10 Постройте таблицы значений для указанных функций.
  1.  $y = x^2 - 5x - 3$ ,  $x \in [-3, 3]$ , вводится значение шага  $h$ .
  2.  $y = 2 + \frac{3x^3 - 7}{x}$ ,  $x \in [a, b]^1$ , вводятся значения  $a$ ,  $b$  и количество точек.

---

<sup>1</sup> Подсказка:  $h = \frac{a+b}{k-1}$ .

**11** Выполните задания для примера 19.10.

1. Команду `writeln('в числе ', k, ' цифр')` заменили командой `writeln('число ', n, ' состоит из ', k, ' цифр')`. Какой результат будет получен и почему? Какие изменения нужно внести в программу для получения правильного результата?

2. Изменится ли результат работы программы, если вместо условия цикла  $n < 0$  использовать условие  $n > 1$ ?

3. Проверьте работу программы для  $n = 0$ . Почему получился такой результат? Что нужно изменить в программе для получения правильного результата?

**12** Программу из примера 19.10 изменили. Сформулируйте задачу, которая решается с помощью данной программы.

```
var i,k,n,z: integer;
begin
  write('Введите n = ');
  read(n);
  write('Введите i = ');
  read(i);
  k:=0;
  while n > 0 do
  begin
    z := n mod 10; //Текущая цифра
    k := k + 1;
    if k = i then
      writeln('В разряде ', i, ' стоит цифра ', z);
    N := n div 10; //Уменьшение числа в 10 раз
  end;
  if i > k then
    writeln('В числе ', k, ' цифр, в разряде ', i, ' нет цифр')
  else
    writeln('В числе ', k, ' цифр');
  end.
```

**13** Дано натуральное число  $n$ . Определите, каких цифр в числе больше — четных или нечетных.

**14** Дано натуральное число  $n$ . Выведите номера разрядов, в которых стоят цифры, кратные 3, или сообщение, что таких цифр нет.

**15** Задано натуральное число. Напишите программу, которая для числа с нечетным количеством цифр выведет на экран цифру, стоящую на средней позиции числа.

Если в числе четное количество цифр, то вывести соответствующее сообщение. Например, для числа 23 452 ответом будет 4, а для числа 56 — сообщение «В числе четное количество цифр».

**16** Число является палиндромом, если оно одинаково читается слева направо и справа налево. Напишите программу, которая по введенному натуральному числу определит, является оно палиндромом или нет. Примеры: 12321, 6776 — палиндромы, 12335 — нет.

**17\*** Натуральное число  $n$  называется числом Армстронга, если сумма цифр числа, возведенных в  $n$ -ю степень, где  $n$  — количество цифр в числе, равна самому числу. Например,  $153 = 1^3 + 5^3 + 3^3$ . Напишите программу, которая найдет:

1. Все трехзначные числа Армстронга.
2. Все четырехзначные числа Армстронга.

**18** В 1626 г. индейцы продали остров Манхэттен за 20 долларов. Если бы эти деньги были помещены в банк на текущий счет и ежегодный прирост составлял бы  $x\%$ , какова была бы стоимость капитала в этом году? Напишите программу, которая ответит на данный вопрос.

**19** Имеются два сосуда. В первом находится  $C_1$  л воды, а во втором  $C_2$  л воды. Из первого сосуда переливают половину воды во второй, а затем из второго переливают половину воды в первый (одно переливание) и т. д. Напишите программу, которая определит, сколько воды окажется в каждом из сосудов после  $k$  переливаний.

**20** Дано натуральное число  $n$ . Написать программу, которая выведет все числа, взаимно простые с  $n$ , а также числа, которые меньше его.

**21\*** Измените программу из примера 19.13:

1. Найти НОД четырех чисел.
2. Найти НОК (наименьшее общее кратное) двух чисел.
3. Вводятся числитель и знаменатель правильной дроби. Сократите дробь.
- 4\*. Две правильные дроби заданы своими числителями и знаменателями. Найдите их сумму. Ответ выведите в виде смешанной дроби.