# 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.

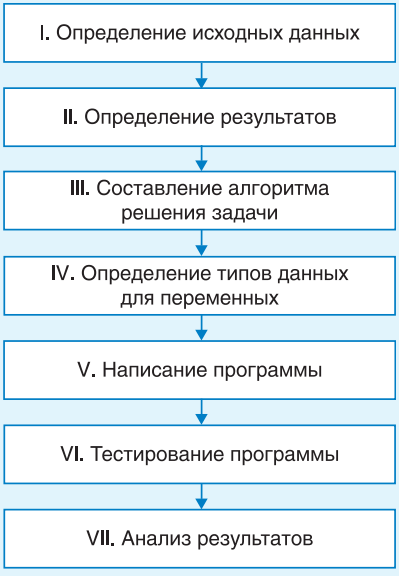
Все команды исполнителя делят на группы:

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.

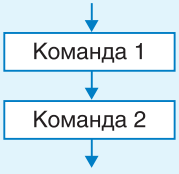
Для поиска ошибок можно использовать средства отладки программ (см. *Приложение 3*, с. 161—162).

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.

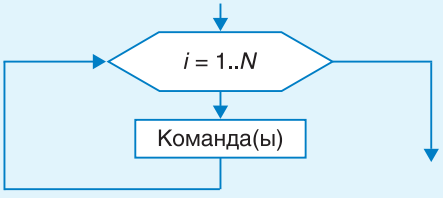
**Пример 13.2.** Блок-схемы алгоритмических конструкций:

1. Следование:

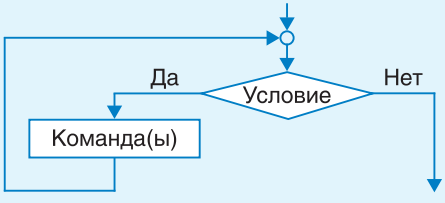



2. Цикл:

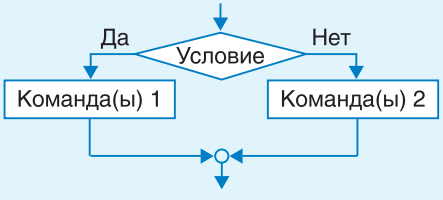
1) цикл с параметром (значение параметра изменяется от 1 до $N$)




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 выводить 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.
```

VI. Тестирование программы:

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

Окно вывода

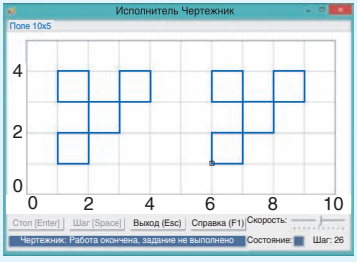
```
Введите x = 3.8
Введите y = 2.7
a = 0.585024492652204
```

VII. Правильность вычислений проверить на калькуляторе.

В примере алгоритмическая конструкция *следование* образована командами:

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

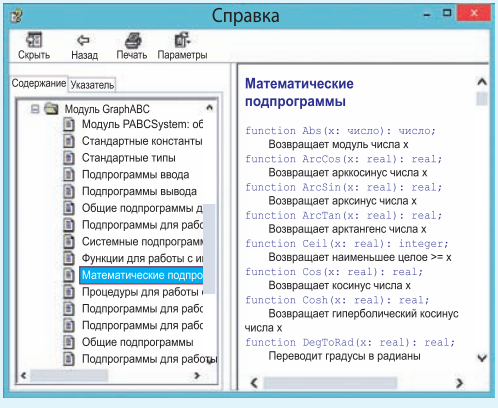
**Пример 13.4.** Написать программу для вывода изображения:




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 |
|---|---|
| `abs(x)` | Находит модуль числа $x$ |
| `sqr(x)` | Возводит число $x$ в квадрат |
| `sqrt(x)` | Находит корень квадратный из числа $x$. Результат — всегда число типа `real` |

| Запись на языке Pascal | Описание |
|---|---|
| `trunc(x)` | Находит целую часть действительного числа $x$ (`real`). Результат — число типа `integer` |
| `frac(x)` | Находит дробную часть действительного числа $x$ (`real`). Результат — число типа `real` |
| `sin(x)` | Вычисляет синус числа $x$. Число $x$ задается в радианах[1] |
| `cos(x)` | Вычисляет косинус числа $x$. Число $x$ задается в радианах |
| `RadToDeg(x)` | Переводит радианы в градусы |
| `DegToRad(x)` | 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?

[1] Радиан — единица измерения углов в Международной системе единиц (1 радиан соответствует величине развернутого угла 180°).

## 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.

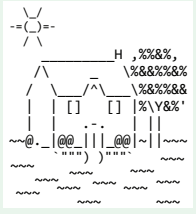
# § 14. Графические возможности среды программирования PascalABC

## 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;`

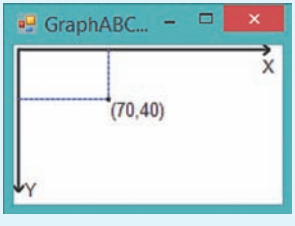
Положение фигур задается координатами в графическом окне. Координатная плоскость в нем отличается от той, которую вы используете на уроках математики. Началом координат является верхний левый угол графического окна — точка (0; 0) (пример 14.1). Координаты задают порядковый номер пикселя по горизонтали 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 × 480 пикселей.



Первые компьютеры не имели возможностей работы с графикой. На печатающих устройствах выводились «картинки», состоящие из символов[1].

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 × 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).

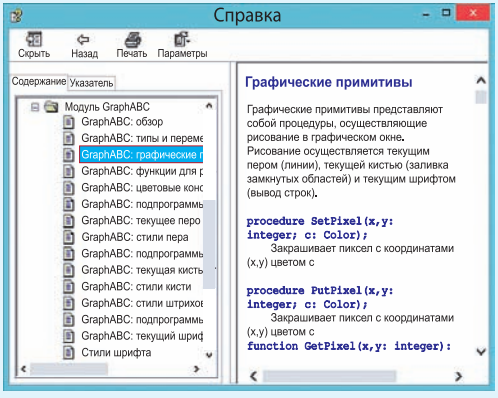



Точка расположена на расстоянии 70 пикселей от левого края окна и на расстоянии 40 пикселей от верхнего края.

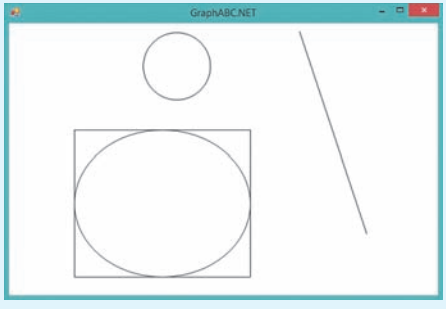
Размеры графического окна можно задать командой `SetWindowSize(n,m);` В скобках указаны размеры окна по горизонтали и вертикали.

[1] https://www.asciiart.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 |
|---|---|
| SetPenColor | Цвет линий |
| SetPenWidth | Толщина линии |
| SetPenStyle | Стиль линий |
| SetBrushColor | Цвет заливки |
| SetBrushStyle | Стиль заливки |
| SetBrushHatch | Вид штриховки для заливки |

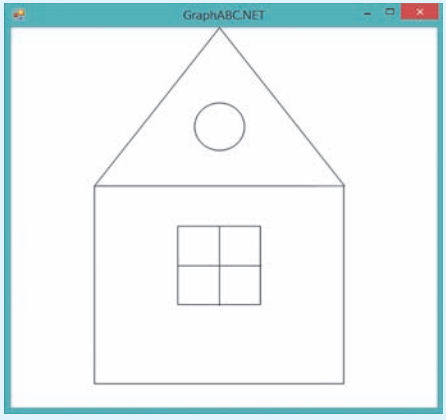
**Пример 14.4.**

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:

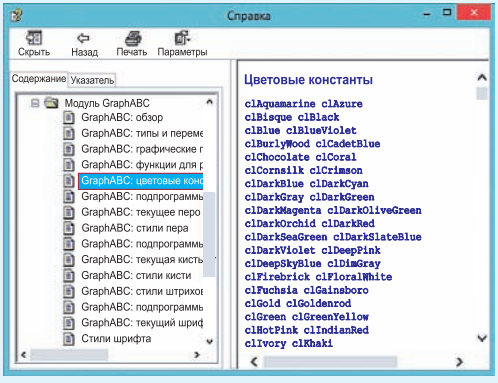



Обратите внимание, что при написании команд у вас появляются подсказки:

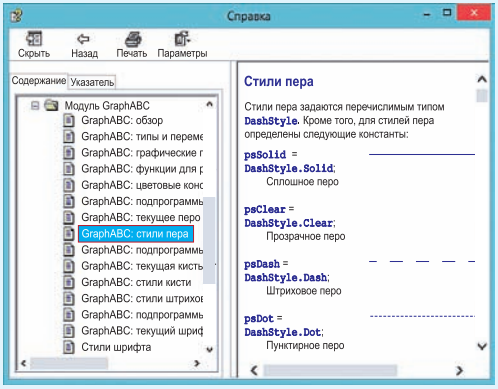

Подсказка появляется также при наведении указателя мыши на уже написанную команду.

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

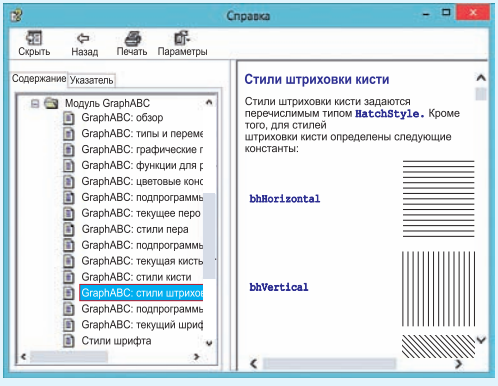
**Пример 14.5.** Цветовые константы.




**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.

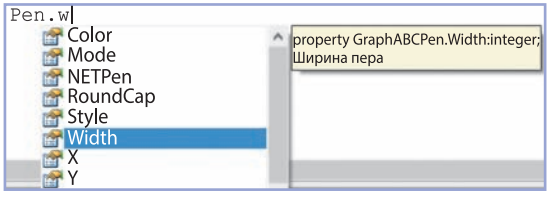
Для фигур по умолчанию установлена заливка белым цветом. Если цвет кисти выбрать до рисования фигуры, то фигура будет закрашена установленным цветом. Цвет уже нарисованной фигуры можно изменить с помощью команды заливки:

`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).

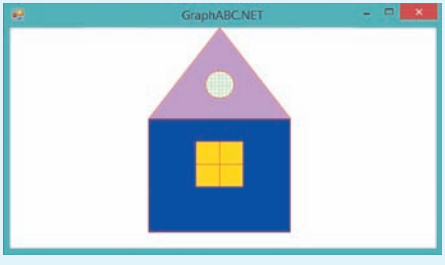
В графическом режиме PascalABC можно выводить в графическое окно тексты и числа.

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

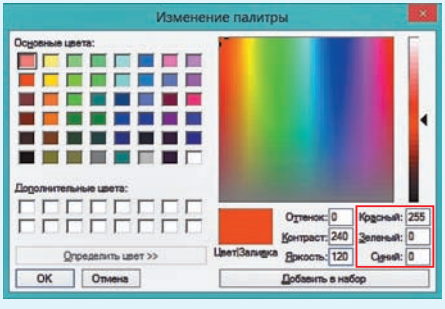
**Пример 14.8.** Программа:

```
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:

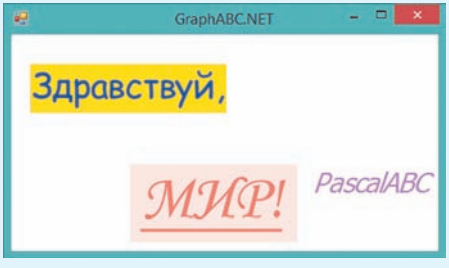


**Пример 14.10.** Выведем в графическое окно приветствие миру, используя разные свойства текста.

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).

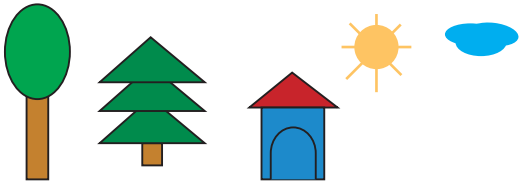
1. Какая библиотека используется для подключения графики в PascalABC?
2. Как определена система координат в графическом окне?
3. Как задать размеры графического окна?
4. Как найти описание графических примитивов в справочнике?
5. Как можно изменить цвет линий, заливки?
6. Как вывести текст в графическом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

### Упражнения

**1** Подпишите изображение из примера 14.8.

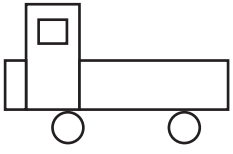
**2** Дополните изображение домика из примера 14.8 изображениями трубы и дыма из трубы в виде нескольких овалов:

**3** Дополните результат, полученный при выполнении задания 2, какими-либо из предложенных изображений или придумайте свои.

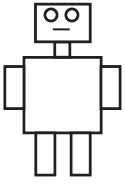


**4** Напишите программу для создания изображения. Раскрасьте данное изображение по своему усмотрению. Дополнительные команды для построения графических примитивов можно найти в справочной системе.

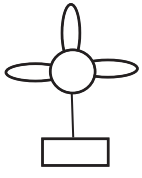
Грузовик
1



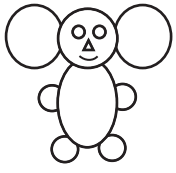
Робот
2



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.