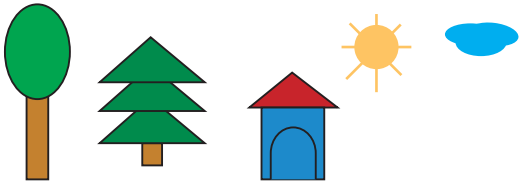
Упражнения

1 Подпишите изображение из примера 14.8.

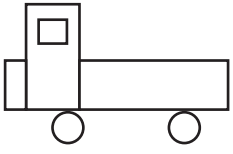
2 Дополните изображение домика из примера 14.8 изображениями трубы и дыма из трубы в виде нескольких овалов:

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

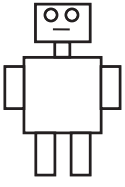


4 Напишите программу для создания изображения. Раскрасьте данное изображение по своему усмотрению. Дополнительные команды для построения графических примитивов можно найти в справочной системе.

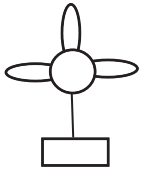
Грузовик
1



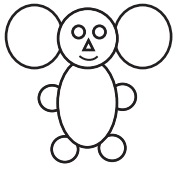
Робот
2



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.

Правообладатель Народная асвета

Пример 15.1. Примеры логических выражений:

- `3 < 7` — логическое выражение, значение которого `true`;
- `2 + 2 * 2 = 8` — логическое выражение, значение которого `false`;
- `abs(-5) > abs(3)` — логическое выражение, значение которого `true`;
- `y >= 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 `false < 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 (≠)	<>
Больше (>)	>
Меньше (<)	<
Больше или равно (⩾)	>=
Меньше или равно (⩽)	<=

Сравнивать 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.

Правообладатель Народная асвета

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 **mod** 2 = 0.

3. Вывод результата.

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

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
```

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*	**not** *A*	*A* **and** *B*	*A* **or** *B*
True	True	False	True	True
False	True	True	False	True
True	False	False	False	True
False	False	True	False	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* **xor** *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.

Правообладатель Народная асвета

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

a **or** (c < b) **and** d

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

Пример 15.4*. Рассмотрим выражение:

a < b **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) **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 **and** c, а затем сравнения.

Пример 15.5. Построение отрицаний:

not not a = a;
not (a **and** b) = (**not** a) **or** (**not** b);
not (a **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.

Правообладатель Народная асвета

2. Вычисление значения логической переменной. Рассмотрим два случая.

Верно неравенство: $A < B < C$. Этому неравенству соответствует логическое выражение: (A < B) **and** (B < C). Присвоим переменной r1 значение этого выражения.

Верно неравенство: $A > B > C$. Этому неравенству соответствует логическое выражение: (A > B) **and** (B > C). Присвоим 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}
```

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).

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.

Правообладатель Народная асвета

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;
```

Правообладатель Народная асвета

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.
```

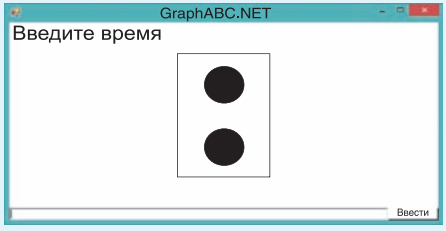
[1] При вводе данных в графическом окне они не отображаются на экране. Для того чтобы видеть, что ввели, необходимо дополнительно вывести введенное значение.

Правообладатель Народная асвета

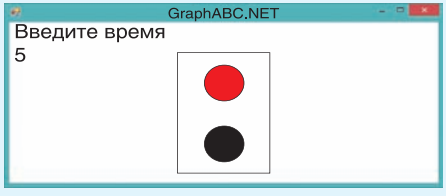
Пример 16.2. *Продолжение.*

VI. Тестирование.

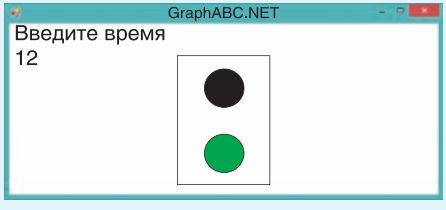
Вид графического окна до ввода числа:



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.

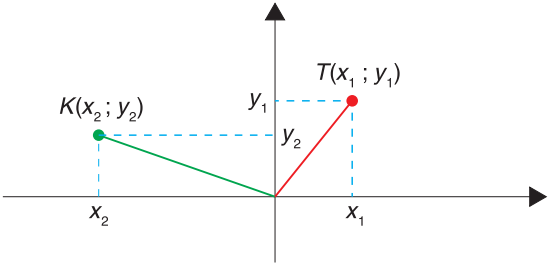
III. Алгоритм решения задачи.

Правообладатель Народная асвета

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:

$$r\_T = \sqrt{x_1^2 + y_1^2} \quad \text{и} \quad r\_K = \sqrt{x_2^2 + y_2^2}.$$



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.

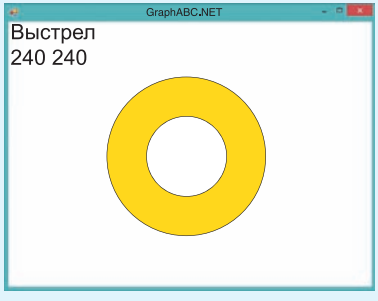
III. Алгоритм решения задачи.

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.

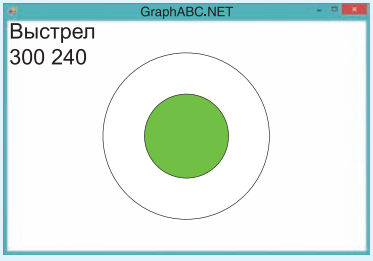
Запустить программу и ввести координаты выстрела (240; 240). Результат:



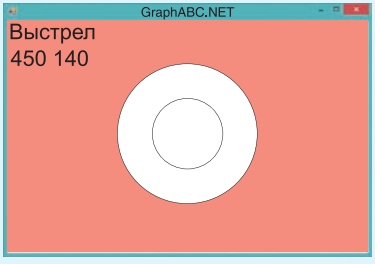
Правообладатель Народная асвета

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 $99 < a < 1000$.

Правообладатель Народная асвета

3. Если число трехзначное, выделяем его цифры:

1) для выделения первой цифры $a1$ находим целую часть от деления числа a на 100;

2) для выделения второй цифры $a2$ числа a находим остаток от его деления на 100, а затем целую часть от деления полученного остатка на 10;

3) последняя цифра числа $a3$ 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.

Правообладатель Народная асвета

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, Юля нашла другой способ вычисления второй цифры трехзначного числа для примера 16.5. Какую из команд использовала Юля? 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?

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

Правообладатель Народная асвета

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 $R1$, $R2$, $R3$ (известно, что $R1 < R2 < R3$). 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.

Правообладатель Народная асвета

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.

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

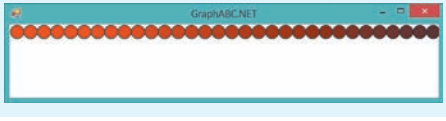
Запустить программу. Результат:

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.
```



Правообладатель Народная асвета

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 var[1] 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.

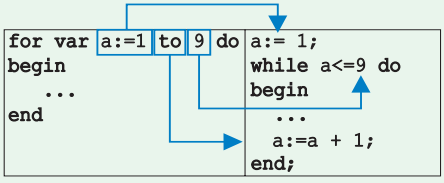
1. Значение переменной a изменяется в цикле от 1 до 9.

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.

[1] Ключевое слово var может быть опущено, тогда переменная *i* должна быть описана (как integer) в разделе описания var до начала программы.

Правообладатель Народная асвета

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.

Правообладатель Народная асвета

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) or (i mod x = 0)`. Тогда условие not `((i < n) or (i mod x = 0))` будет истинным. Согласно правилам построения отрицаний (см. пример 15.5) это условие можно заменить условием: `(i >= n) and (i mod x <> 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.

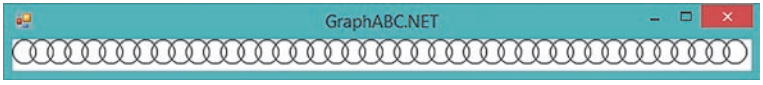
1. Радиусы окружностей равны 20.
2. Окружности располагаются вдоль левого края окна.
3. Радиус окружности вводится пользователем.

Правообладатель Народная асвета

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).

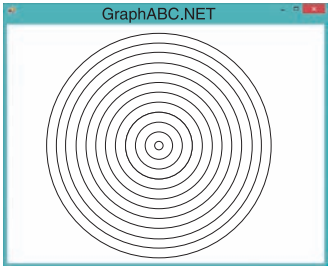
5 Какие изменения нужно внести в программу из примера 17.3, чтобы получить следующее изображение? Функции для изменения интенсивности цвета см. в примере 17.1.

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*.

Правообладатель Народная асвета