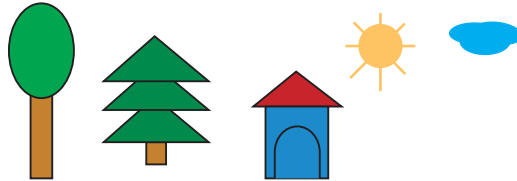




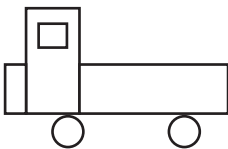
Упражнения

- 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.

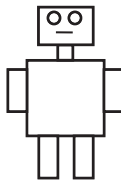


- 4 Напишите программу для создания изображения. Раскрасьте данное изображение по своему усмотрению. Дополнительные команды для построения графических примитивов можно найти в справочной системе.

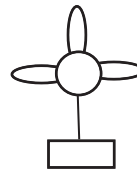
Грузовик
1



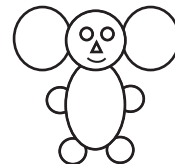
Робот
2



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 получают-ся 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	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

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 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.

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.

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 $r1 \text{ or } r2$.

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:

$D := \text{Pred}(\text{true}); \quad \{D = \text{false}\}$

$E := \text{Succ}(\text{false}); \quad \{E = \text{true}\}$



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;
```