Логический Тип данных. Условный оператор.

Булева алгебра

Двоичное кодирование – все виды информации кодируются с помощью 0 и 1.

Задача – разработать оптимальные правила обработки таких данных.

Джордж Буль разработал основы алгебры, в которой используются только 0 и 1 (алгебра логики, булева алгебра).

Почему «логика»?

Результат выполнения операции можно представить как истинность (1) или ложность (0) некоторого высказывания.

Логические высказывания

Логическое высказывание – это повествовательное предложение, относительно которого можно однозначно сказать, истинно оно или ложно.

Высказывание или нет?

- Сейчас идет дождь.
- Жирафы летят на север.
- История – интересный предмет.
- У квадрата – 10 сторон и все разные.
- Красиво!
- В городе N живут 2 миллиона человек.
- Который час?

В Паскале логические значения обозначаются служебными словами false (ложь) и true (истина), а идентификатор логического типа — boolean.

Var a: boolean;

Логические операции

- Операция НЕ (инверсия)
- Операция ИЛИ (логическое сложение, дизъюнкция)
- Операция И (логическое умножение, конъюнкция)
- Операция «исключающее ИЛИ»
- Операции отношений

Операция НЕ (инверсия)

Если высказывание **А** истинно, то «**не А**» ложно, и наоборот.

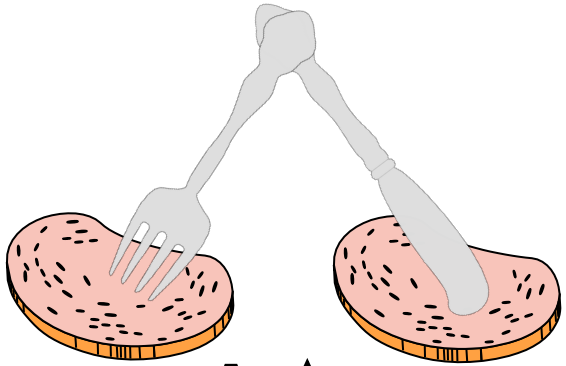
А	не А
0	1
1	0

Таблица истинности логического выражения Х – это таблица, где в левой части записываются все возможные комбинации значений исходных данных, а в правой – значение выражения Х для каждой комбинации.

Операция И (логическое умножение, конъюнкция)

Высказывание «**A и B**» истинно тогда и только тогда, когда **A** и **B** истинны одновременно.

	A	B	A и B
0	0	0	0
1	0	1	0
2	1	0	0
3	1	1	1



A ∧ B

конъюнкция – от лат. *conjunctio* — соединение

Операция ИЛИ (логическое сложение, дизъюнкция)

Высказывание «**A или B**» истинно тогда, когда истинно **A** или **B**, или оба вместе.

A	B	A или B
0	0	0
0	1	1
1	0	1
1	1	1

дизъюнкция – от лат. *disjunctio* — разъединение

Операция «исключающее ИЛИ»

Высказывание «**A ⊕ B**» истинно тогда, когда истинно **A** или **B**, но ***не оба одновременно***.

A	B	A ⊕ B
0	0	0
0	1	1
1	0	1
1	1	0

сложение по модулю 2: $A \oplus B = (A + B) \bmod 2$

A	*B*	Not *A*	*A* And *B*	*A* Or *B*	*A* Xor *B*
T	T	F	T	T	F
T	F	F	F	T	T
F	F	T	F	F	F
F	T	T	F	T	T

1. Вычислить значения логических выражений:

а)K mod 7=K div 5-1 при K=15;

б)odd(trunc(10*P)) при P=0.182;

В)not odd(n)при n=0;

г)t and (P mod 3=0) при t=true, P=10101;

Д) (x*y<>0) and (y>x) при x=2, y=I;

е) a or not b при a=false, b=true.

2. Если a=true и x=1, то какое значение получит логическая переменная dпосле выполнения оператора присваивания:

а) d:=x<2; б) d:=not a or odd(x);

Разветвляющиеся алгоритмы

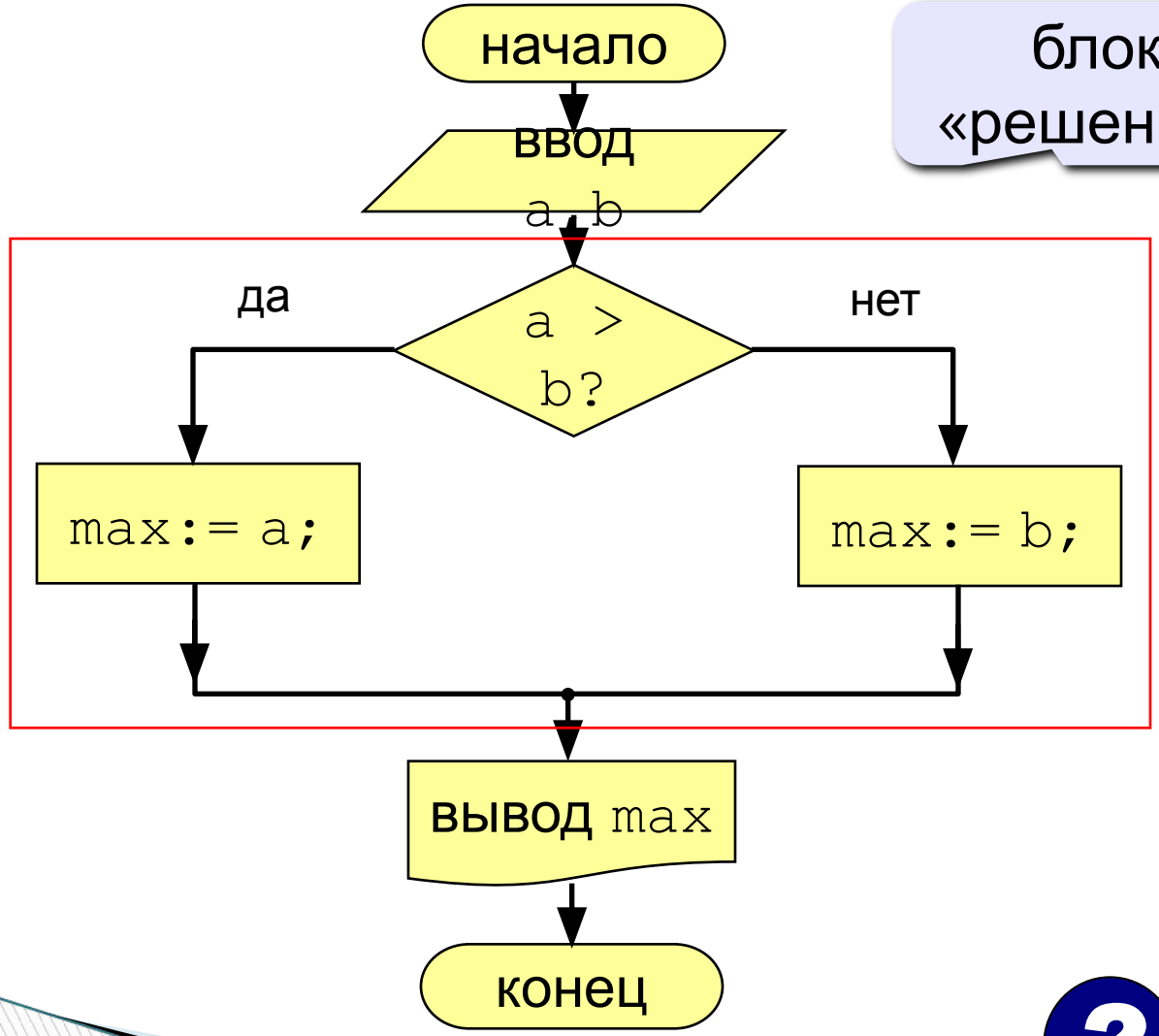
Задача. Ввести два целых числа и вывести на экран наибольшее из них.

Идея решения: надо вывести на экран первое число, если оно больше второго, или второе, если оно больше первого.

Особенность: действия исполнителя зависят от некоторых условий (*если … иначе …*).

Алгоритмы, в которых последовательность шагов зависит от выполнения некоторых условий, называются разветвляющимися.

Вариант 1. Блок-схема



блок «решение»

полная форма ветвления

? Если a = b?

Вариант 1. Программа

```
program qq;
var a, b, max: integer;
begin
   writeln('Введите два целых числа');
   read ( a, b );
   if a > b then begin
     max := a;
   end
   else begin
     max := b;
   end;
   writeln ('Наибольшее число ', max);
end.
```

полная форма условного оператора

Условный оператор

```
if <условие> then begin
   {что делать, если условие верно}
end
else begin
   {что делать, если условие неверно}
end;
```

Особенности:

- перед *else* НЕ ставится точка с запятой
- вторая часть (*else* …) может отсутствовать (неполная форма)
- если в блоке один оператор, можно убрать слова *begin* и *end*

Что неправильно?

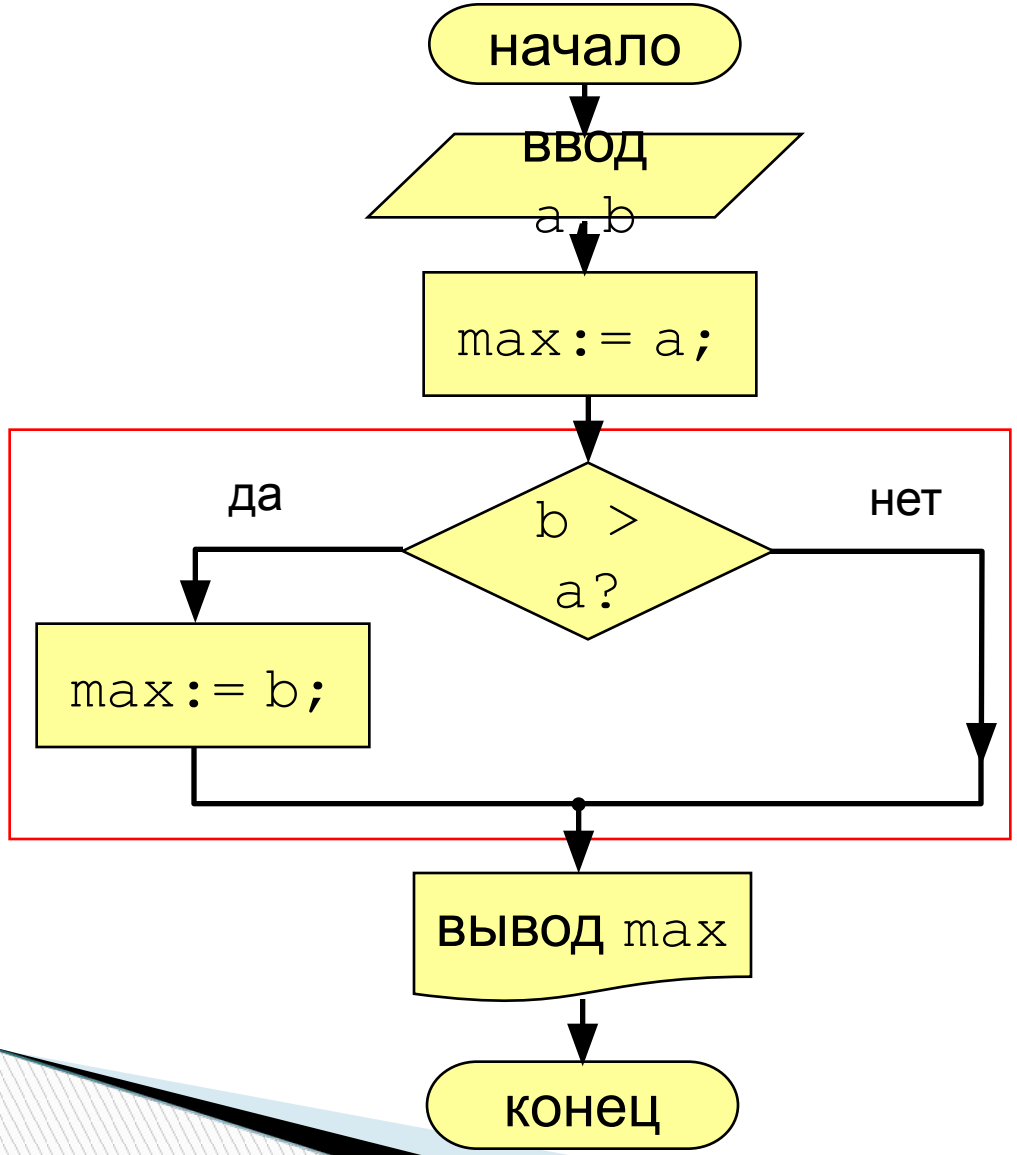
```
if a > b then begin
   a := b;
end
else begi
n
   b := a;
end;
```

```
if a > b then begin
   a := b; end
else begin
   b := a;
end;
```

```
if a > b then begin
   a := b;
end
else begin
   b := a;
end;
```

```
if a > b then begin
   a := b;
end
 else begin
   b := a;
end;
```

Вариант 2. Блок-схема



неполная форма ветвления

Вариант 2. Программа

```
program qq;
var a, b, max: integer;
begin
   writeln('Введите два целых числа');
   read ( a, b );
   max := a;
   if b > a then
     max := b;
   writeln ('Наибольшее число ', max);
end.
```

неполная форма условного оператора

Вариант 2Б. Программа

```
program qq;
var a, b, max: integer;
begin
   writeln('Введите два целых числа');
   read ( a, b );
   max := b;
   if a > b then
      max := a;
   writeln ('Наибольшее число ', max);
end.
```

Что неправильно?

```
if a > b then
      a := b
else b := a;
```

```
if a > b then begin
   a := b;
end
else b := a;
```

```
if a > b then
      a := b
else b := a;
```

```
if b >= a then
   b := a;
```