



Ветвления в алгоритмах


Pascal

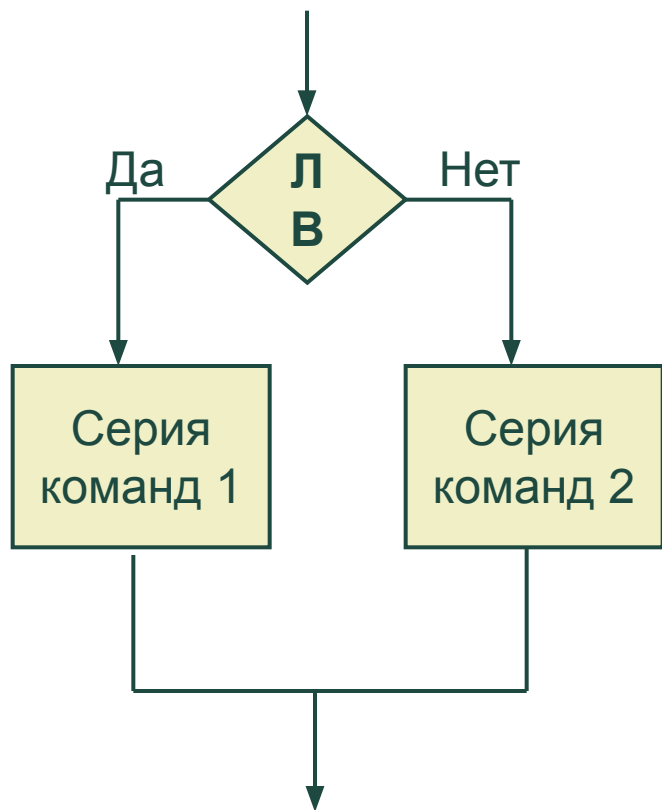
О чем пойдет речь

- ❖ Понятие ветвления в алгоритмах
- ❖ Команды ветвления
- ❖ Построение логических выражений
- ❖ Реализация команд ветвления в языке Паскаль

- ❖ **Ветвление** – это такая форма организации действий, при которой в зависимости от выполнения или невыполнения некоторого условия совершается одна или другая последовательность команд.
- ❖ **Разветвляющиеся алгоритмы** – алгоритмы, содержащие команду ветвления .
- ❖ **Логическое выражение (ЛВ)** – это высказывание (утверждение), относительно которого можно однозначно сказать, **истинно** оно или **ложно**. Логическое выражение часто называют *условием*.
- ❖ Команду ветвления чаще всего называют командой **если** по первому слову конструкции команды.

Варианты команды ветвления

Вариант 1 (полный)



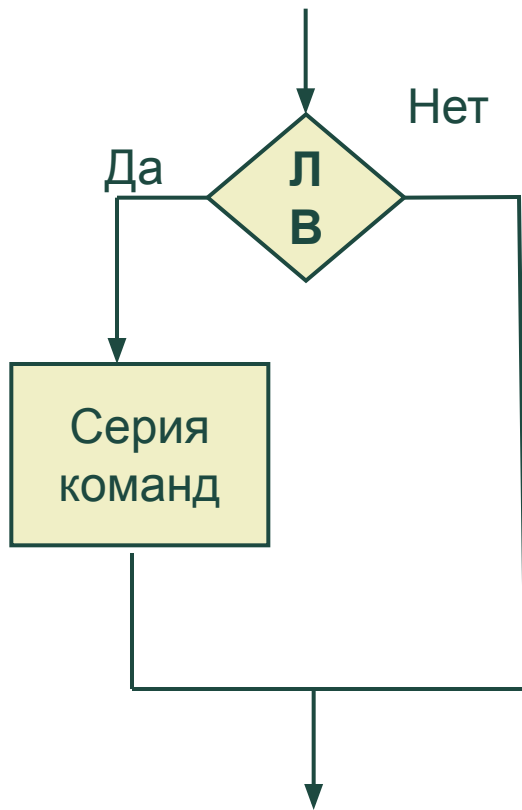
Если *<логическое выражение>*
то *<серия команд 1>*
иначе *<серия команд 2>*
Конец ветвления

Реализация команды на языке Паскаль

If *<логическое выражение>*
Then
 Begin *<Серия команд 1>* **end**
Else
 Begin *<Серия команд 2>* **end;**

Варианты команды ветвления

Вариант 2 (неполный)



Если *<логическое выражение>*
то *<серия команд>* **>**
Конец ветвления

**Реализация команды
на языке Паскаль**

If *<логическое выражение>*
Then
Begin *<Серия команд>* **end;**

- ❖ **Выражением отношения** называется словосочетание языка, в котором два выражения связаны знаком операции отношения. Выражение отношения определяет *истинность* или *ложность* результата.
- ❖ **Операции отношения** выполняют сравнение двух операндов и определяют, *истинно* значение выражения или *ложно*. Результат операции отношения всегда принимает одно из двух значений: **True** (истина) или **False** (ложь).
- ❖ **Операнд** – величина или выражение, над которыми производят операцию.

Операции отношения

Операция	Название	Выражение	Результат
=	равно	$A = B$	True, если $A = B$
<>	не равно	$A <> B$	True, если $A <> B$
>	больше	$A > B$	True, если $A > B$
<	меньше	$A < B$	True, если $A < B$
>=	больше или равно	$A >= B$	True, если $A >= B$
<=	меньше или равно	$A <= B$	True, если $A <= B$

NOT – логическое отрицание. Формат записи: **not A**

Выражение	A	Результат
Not A	True	False
	False	True

Примечание: A – простое логическое выражение (условие).

Логические операции

AND – логическое **И**. Формат записи: **A and B**

Выражение	A	B	Результат
A and B	True	True	True
	True	False	False
	False	True	False
	False	False	False

Примечание: A и B – простые логические выражения (условия).

OR – логическое **ИЛИ**. Формат записи: A **or** B

Выражение	A	B	Результат
A or B	True	True	True
	True	False	True
	False	True	True
	False	False	False

Примечание: A и B – простые логические выражения (условия).

XOR – исключающее **ИЛИ**. Формат записи: **A xor B**

Выражение	A	B	Результат
A xor B	True	True	False
	True	False	False
	False	True	False
	False	False	True

Примечание: A и B – простые логические выражения (условия).

Порядок выполнения логических операций

Операция	Приоритет
NOT	Первый
AND	Второй
OR, XOR	Третий
=, <>, >=, <=	Четвертый

Приоритетом называют очередность выполнения операций в выражении. Выполнение каждой операции происходит с учетом ее приоритета.

Правила определения старшинства операций

- ❖ Операнд, находящийся между двумя операциями с *различными* приоритетами, связывается с операцией, имеющей более *высокий* приоритет.
- ❖ Операнд, находящийся между двумя операциями с *равными* приоритетами, связывается с операцией, которая находится *слева*.
- ❖ Выражение, заключенное в скобки, перед выполнением вычисляется как отдельный операнд.
- ❖ Операции с *равными* приоритетами производятся *слева направо* с возможным регулированием порядка выполнения скобками.

Примеры логических выражений

Выражение	Результат
(3>2) and (21<>100)	True
(3>2) or (61=100)	True
not (23>15)	False
(3>2) xor (61<>100)	False
not (23<15) and (23<>0)	True
(23<>25) or (23<13) and (3<1)	True

Вложение команд ветвления друг в друга

Если *<логическое выражение 1>* **то**
 Если *<логическое выражение 2>*
 то *<серия команд 1>*
 иначе *<серия команд 2>*
иначе *<серия команд 3>*
Конец ветвления

Примечание: При вложении команд ветвления следует иметь в виду, что служебное слово **Иначе** всегда связывается с ближайшим по тексту служебным словом **Если**, которое еще не связано со служебным словом **Иначе**.

Вложение условного оператора

```
If <условие 1> Then
    If <Условие 2> Then
        Begin <Серия операторов 1> End
    Else
        Begin <Серия операторов 2> End
Else
    Begin <Серия операторов 3> End;
```

Примечание: При вложении условных операторов следует иметь в виду, что служебное слово **Else** всегда связывается с ближайшим по тексту служебным словом **If**, которое еще не связано со служебным словом **Else**.

Команда ветвления

Границы применимости



Команда ветвления применяется в том случае, если в процессе решения задачи приходится выбирать *из двух взаимоисключающих* (противоположных) условий.

Например: $X \geq 0$ и $X < 0$

Примечание: Логические выражения прописанные в команде ветвления и являются теми *условиями*, при выполнении или не выполнении которых и происходит выбор той или иной серии команд.

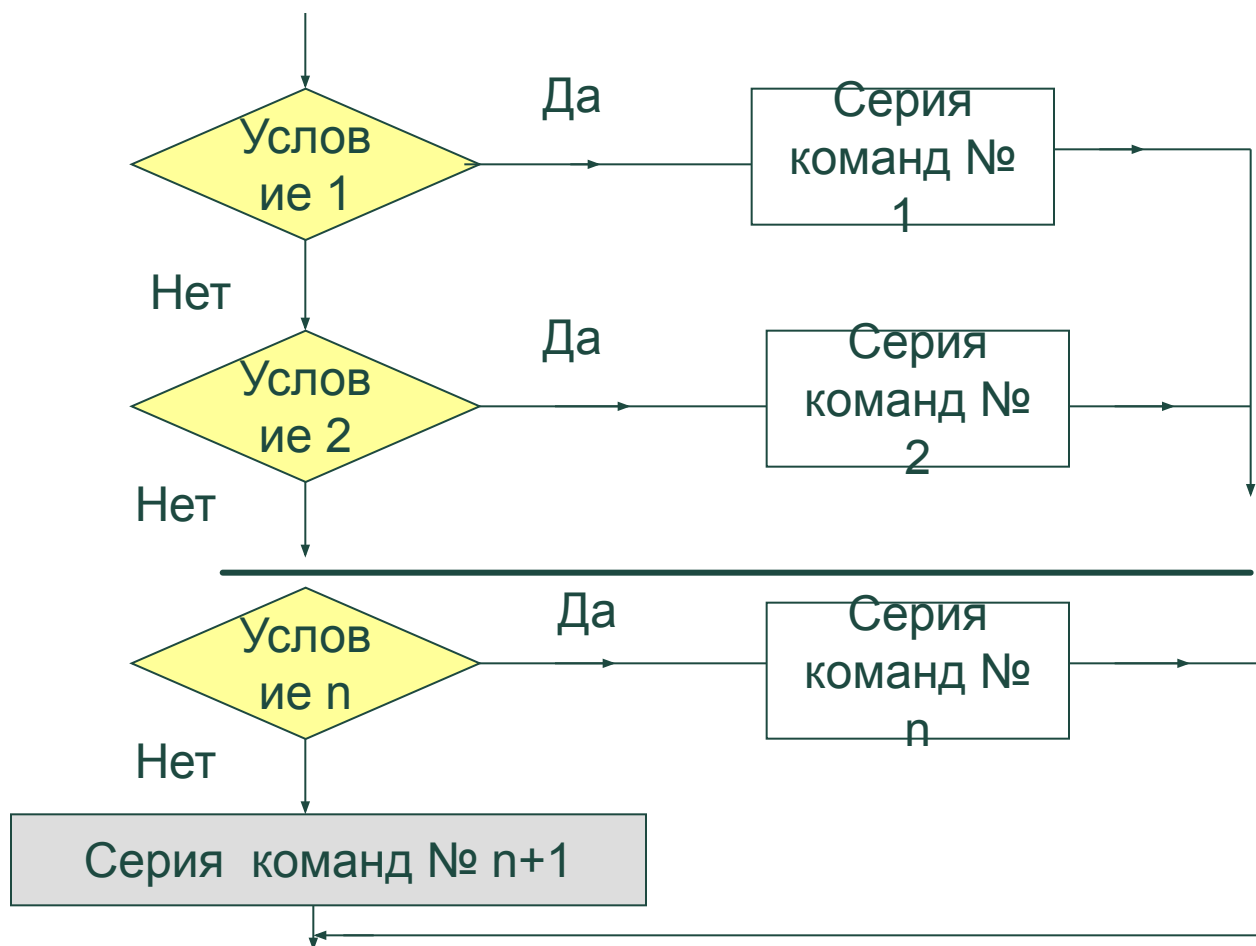
Команда «Выбор»

Команда **«Выбор»** применяется в том случае, когда при решении задачи приходится выбирать *не из двух, а из нескольких* вариантов условий.

Причем варианты условий, также как и в команде ветвления должны *взаимно исключать* друг друга.

Блок-схема команды «Выбор»

Вариант 1



Команда «Выбор» на языке Паскаль

Вариант 1

Case <Выражение-селектор> **of**

Значение 1: **Begin** <Список операторов 1> **end**;

Значение 2: **Begin** <Список операторов 2> **end**;

.....

Значение n: **Begin** <Список операторов n> **end**;

Else

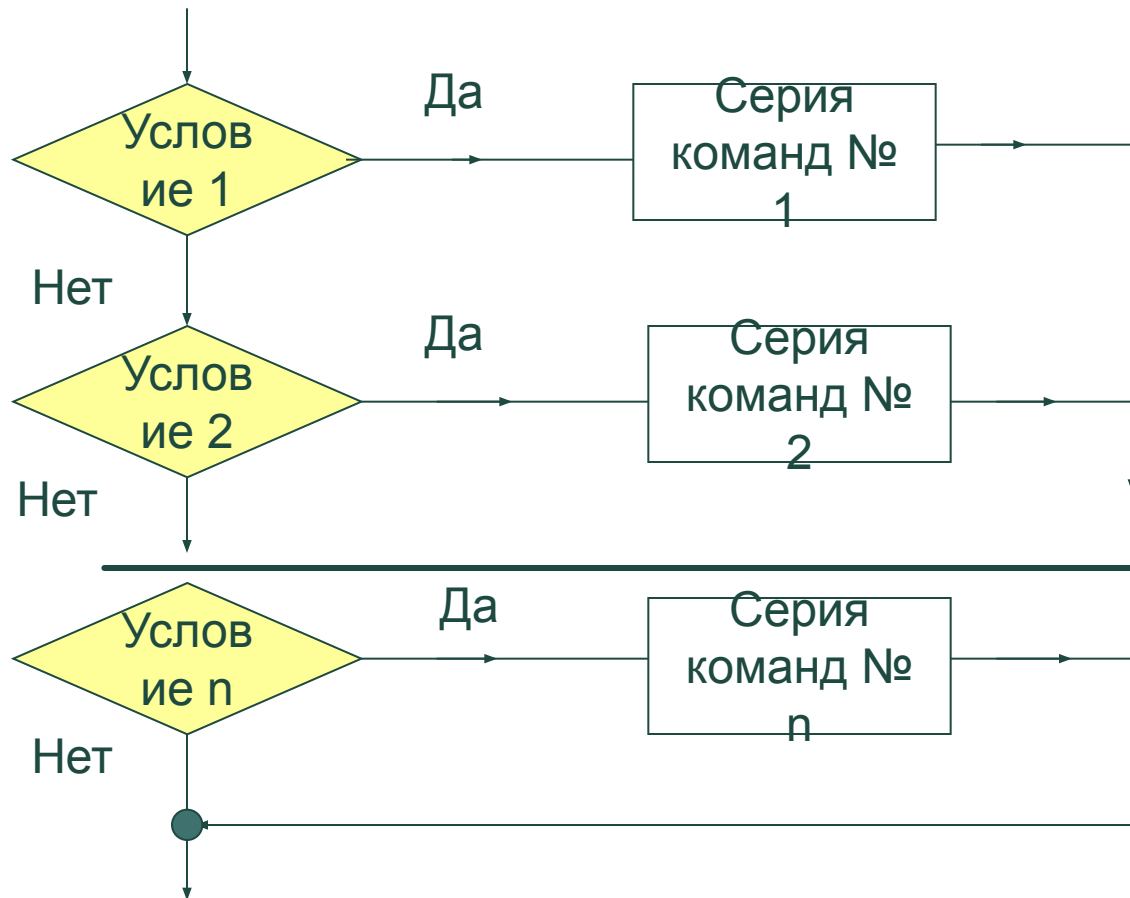
Begin <Список операторов> **end**;

End;

Выражение-селектор – это выражение, по значению которого и происходит переход к тому или иному списку операторов.

Блок-схема команды «Выбор»

Вариант 2



Команда «Выбор» на языке Паскаль

Вариант 2

Case <Выражение-селектор> **of**

Значение 1: **Begin** <Список операторов 1> **end**;

Значение 2: **Begin** <Список операторов 2> **end**;

.....

Значение n: **Begin** <Список операторов n> **end**;

End;

Значения – это константы или диапазон констант дискретного типа.

Правила использования оператора CASE

1. Значения выражения-селектора, записанного после служебного слова **case**, должны принадлежать дискретному типу: **byte**, **integer**, **char** (символьный).
2. Все значения, предшествующие спискам операторов должны иметь тип, *совместимый* с типом выражения-селектора.
3. Все значения в альтернативах должны быть уникальны в пределах оператора **case** (т.е. повторения значений не допускается). Если значения являются диапазонами, то они не должны пересекаться.

Пример. Оператор CASE

Значения интервального типа

```
Program Wozrast;  
Var W: integer;  
Begin  
    Write ('Введите возраст'); Readln (W);  
    Case W of  
        1..6:   Writeln ('Дошкольник');  
        7..17:  Writeln ('Школьник');  
        18..60: Writeln ('Рабочий');  
        61..100: Writeln ('Пенсионер')  
    Else  
        Writeln ('Ошибка при вводе возраста')  
    End;  
End.
```

Пример. Оператор CASE

```
Program Func;  
Var  K: byte;  
     Z: Real;  
Begin  
    Write ('Введите аргумент K'); Readln (K);  
    Case K of  
        1: Z:= K+10;  
        3: Z:= Sgr (K) - 4;  
        18: Z:= K  
    End;  
    Writeln ('Ответ: Z = ', Z)  
End.
```

Пример. Оператор CASE

Значения символьного типа

```
Program Calc;  
Var  X, Y, Rezult: real;  Operation: Char;  
Begin  
    Write ('Введите числа X и Y'); Readln (X,Y);  
    Write ('Введите операцию +, - , *'); Readln  
(Operation);  
    Case Operation of  
        '+' : Rezult:= X + Y;  
        '-' : Rezult:= X - Y;  
        '*' : Rezult:= X * Y  
    End;  
    Writeln ('Ответ:  ',Rezult)  
End.
```