

Программирование  
разветвляющихся алгоритмов на  
языке Паскаль  
Лекция 3

# План

---

1. Оператор условного перехода if
2. Оператор выбора case
3. Оператор безусловного перехода goto

## Литература



# Литература

---

1. Касторнов А.Ф., Евстратова Г.А. Язык программирования Паскаль : учебное пособие для вузов. - Череповец : ГОУ ВПО ЧГУ, 2010. - 117 с. - Библиогр.: С.114.
2. Электронный учебник по языку программирования Паскаль  
[/http://pascal.guti.ru](http://pascal.guti.ru)

# Оператор условного перехода if

---

- Оператор условного перехода if позволяет выполнить одну или другую последовательность действий (операторов) в зависимости от истинности или ложности некоторого условия.
- В повседневной жизни условие обычно формулируется в виде вопроса, на который можно ответить **Да** или **Нет**.

*Например:*

Сумма больше 300?

Номер дня недели равен 7?



# Оператор условного перехода if

---

В программе **условие** – это **выражение логического типа** (Boolean), которое может принимать одно из двух значений: **истина** (True) или **ложь** (False). При описании условий используются:

- операторы сравнения (=, <>, >, <, >=, <=);
- логические операции (NOT, AND, OR, XOR).



# Оператор условного перехода if

---

- Если в условии использован один оператор сравнения, то такое условие называется **простым**. Из простых условий при помощи логических операций можно строить **сложные** (составные) условия. При записи сложных условий необходимо учитывать то, что логические операторы имеют более высокий приоритет, чем операторы сравнения, и поэтому простые условия следует заключать в скобки.



# Оператор условного перехода if

---

*Например:*

Var Summa, Day: Integer;

Summa > 300 {Сумма больше 300? – простое условие}

Day=7 { Номер дня недели равен 7? – простое условие}

(Summa>300) and (Sum<1000) {Сумма больше 300 и меньше 1000 рублей? – составное условие}

(Day=6) Or (Day=7) {Это выходной день? – составное условие}



# Оператор условного перехода if

---

**Условный оператор if работает следующим образом:**

1. Вычисляется значение условия.
2. Если условие истинно (True), то выполняется Оператор1, после чего выполнение оператора if заканчивается. Если условие ложно (False), то выполняется Оператор2, после чего выполнение оператора if заканчивается.

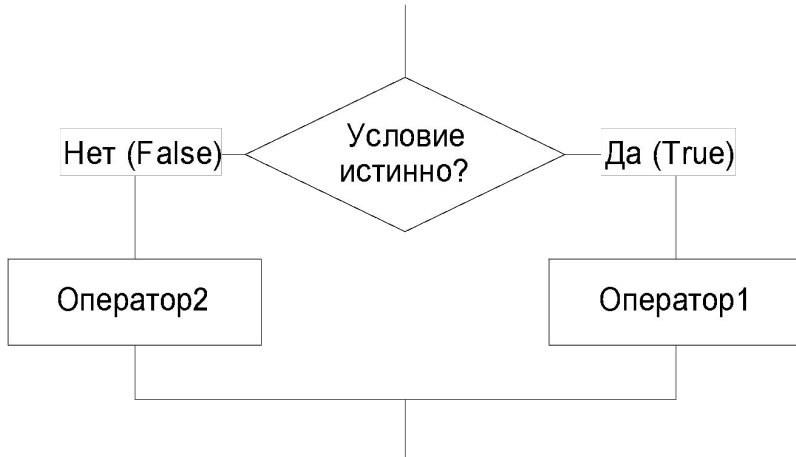




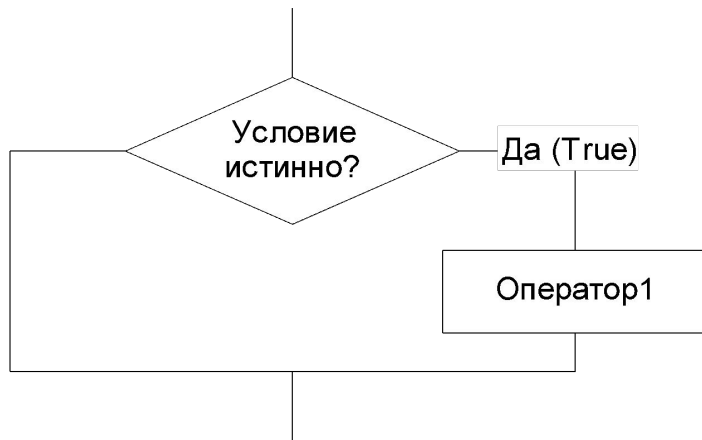
# Оператор условного перехода if

---

□ Полная развилка



□ Неполная развилка. В этом случае, при ложности условия, работа оператора if заканчивается, и никакие действия им не производятся.



# Оператор условного перехода if

---

**Оператор if записывается следующим образом:**

{Полная развилка. Перед служебным словом else «;» не ставится}

```
if условие then Оператор1 else Оператор2;
```

{Неполная развилка}

```
if условие then Оператор1;
```



# Оператор условного перехода if

---

Если в программе по той или иной ветке необходимо выполнить несколько операторов, то запись оператора if выполняется следующим образом:

```
if условие then  
    begin  
        блок операторов 1  
    end  
else  
    begin  
        блок операторов 2  
    end;
```

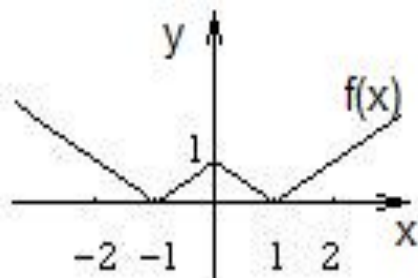


# Оператор условного перехода if

---

## *Пример*

Дано действительное  $x$ . Для функции  $f$ , график которой представлен на рисунке, вычислить  $f(x)$ .



## *Решение задачи*

Математическая модель: функция вычисляется по следующей формуле:

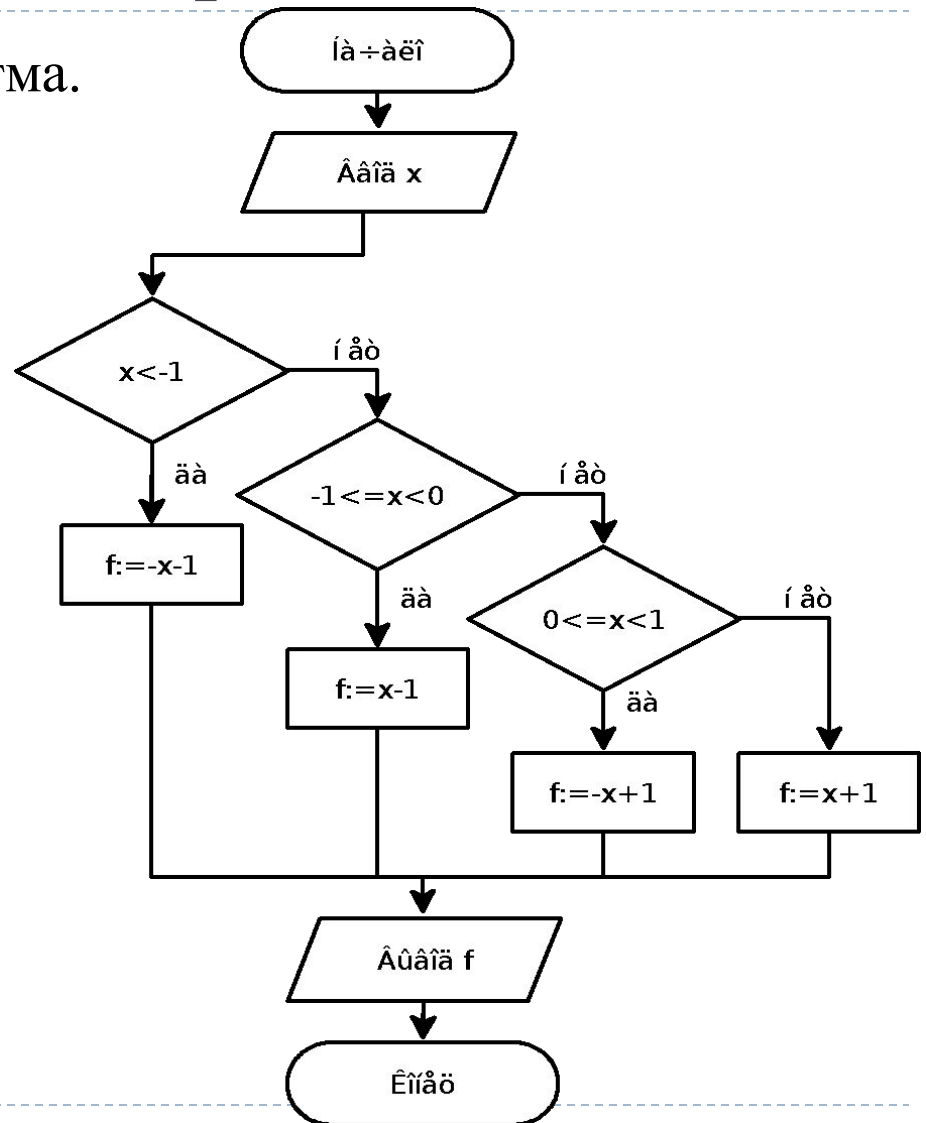
$$f(x) = \begin{cases} -x - 1, & x < -1 \\ x - 1, & -1 \leq x < 0 \\ -x + 1, & 0 \leq x < 1 \\ x + 1, & x \geq 1 \end{cases}$$



# Оператор условного перехода if

□ Составим блок-схему алгоритма.

$$f(x) = \begin{cases} -x-1, & x < -1 \\ x-1, & -1 \leq x < 0 \\ -x+1, & 0 \leq x < 1 \\ x+1, & x \geq 1 \end{cases}$$



# Оператор условного перехода if

---

Переведем алгоритм на язык Паскаль.

```
Program ex1;  
var x, f:Real;  
begin  
  Write('Введите x: ');  
  Readln(x);  
  if x<-1 then f:= -x-1 else  
    if (x>=-1) and (x<0) then f:= x-1 else  
      if (x>=0) and (x<1) then f:= -x+1 else f:= x+1;  Writeln('f=  
,f:6:2);  
  Readln;  
end.
```

# Оператор выбора case

---

- Часто возникают ситуации, когда в программе приходится осуществлять выбор одного из нескольких альтернативных вариантов. Несмотря на то, что такой выбор можно организовать с помощью оператора if, удобнее воспользоваться специальным оператором выбора Case.



# Оператор выбора case

---

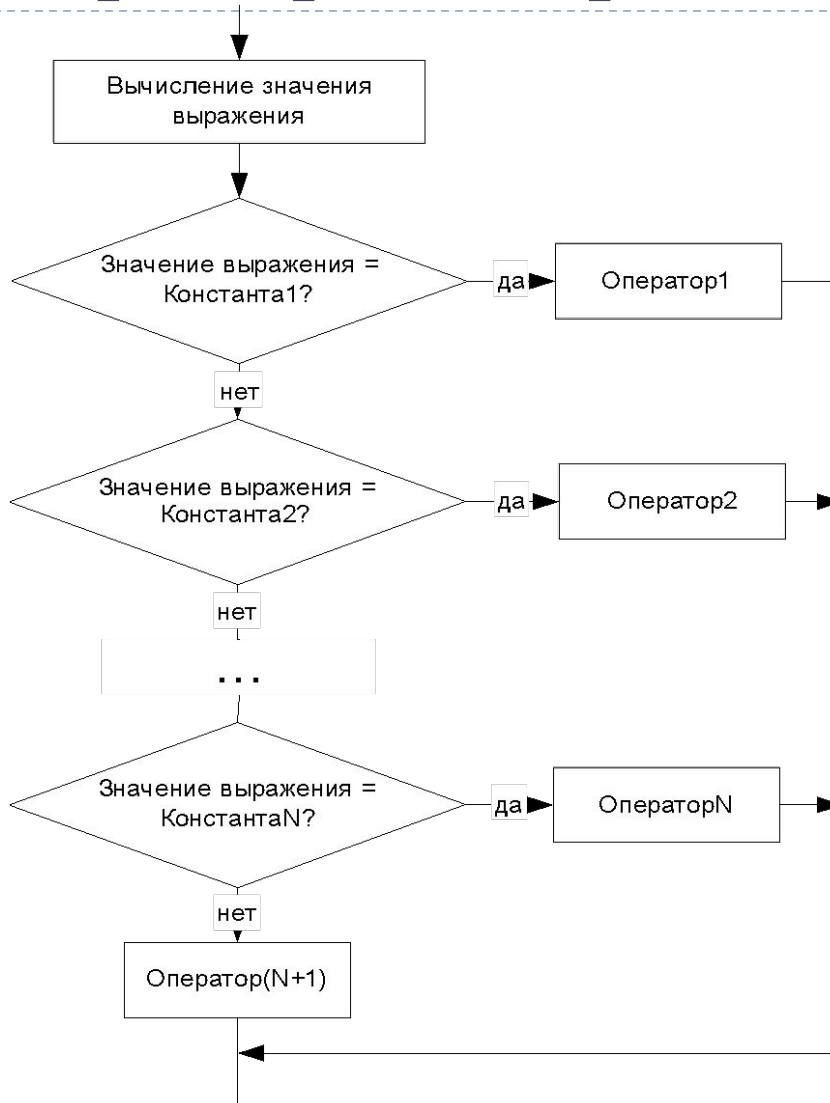
Оператор выбора Case работает следующим образом:

1. Вычисляется значение выражения.
2. Это значение последовательно сравнивается с константами выбора из списка констант.
3. Если значение выражения совпало с константой, то выполняется соответствующий данной константой оператор. На этом выполнение оператора Case заканчивается.
4. Если значение выражения не совпало ни с одной из констант выбора, то выполняется оператор, следующий за служебным словом else. Если ветка else в операторе не описана, то оператор Case никаких действий не производит.





# Оператор выбора case



# Оператор выбора case

---

- Оператор Case может работать только с выражениями порядковых типов. Это, например, типы Integer, Boolean.
- В общем виде оператор Case записывается следующим образом:

```
case Выражение of  
    Константа1: Оператор1;  
    Константа2: Оператор2;  
    ...  
    КонстантаN: ОператорN;  
else Оператор(N+1)  
end;
```

- Перед служебными словами **else** и **end** «;» ставить необязательно. Ветку else в записи оператора можно опускать.



# Оператор выбора case

---

- Если при нескольких константах выбора выполняется один и тот же оператор, то константы перечисляются через запятую, затем ставится «:» и указывается выполняемый оператор.

*Например*

case Выражение of

Константа1, Константа2, Константа3: Оператор;

Константа4: Оператор4;

...

КонстантаN: ОператорN;

else Оператор(N+1)

end;



# Оператор выбора case

---

- Если константы выбора представляют собой диапазон целых чисел, то можно указать первую и последнюю константу диапазона, разделив их двумя точками.

*Например*

1..6

5..8, 10, 15, 17..20



# Оператор выбора case

---

- Если по веткам необходимо выполнить несколько операторов, то запись оператора Case выполняется следующим образом:

```
case Выражение of  
Константа1: begin  
    блок операторов 1;  
end;  
...  
КонстантаN: begin  
    блок операторов N;  
end;  
else begin  
    блок операторов (N+1);  
end;  
end;
```



# Оператор выбора case

---

*Пример:* по номеру дня недели вывести сообщение – рабочий это день или выходной (выходными считаются суббота и воскресенье).

```
program Ex2;  
var Day: integer;  
begin  
  Writeln ('Введите номер дня недели в диапазоне от 1 до 7');  
  Readln (Day);  
  Case Day of  
    1..5: Writeln ('Рабочий день');  
    6..7: Writeln ('Выходной день');  
  Else Writeln ('Номер дня недели введен неверно!');  
end;  
Readln;  
end.
```

# Оператор безусловного перехода goto

---

- Помимо операторов условного перехода существует также оператор безусловного перехода goto.
- Формат:  
`goto метка;`
- Оператор goto переходит при выполнении программы к оператору, отмеченному указанной меткой. Метка должна быть описана в разделе описания меток label.



# Оператор безусловного перехода goto

---

*Пример*

```
label 1;
```

```
...
```

```
begin
```

```
...
```

```
goto 1;
```

```
...
```

```
1: WriteLn('Переход к метке 1');
```

```
...
```

```
end.
```





# Оператор безусловного перехода goto

---

Понятие структурного программирования и общепринятый стиль программирования **НЕ ПРИВЕТСТВУЕТ** применение меток и операторов безусловного перехода в программах. Это затрудняет понимание программы как автором, так и потребителями, кроме того, применение меток отрицательно сказывается на эффективности генерируемого кода.

