

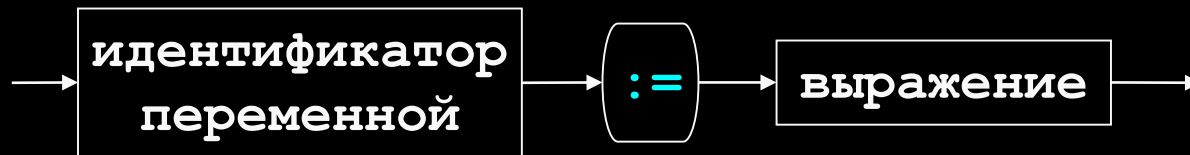
Глава 6. УПРАВЛЯЮЩИЕ СТРУКТУРЫ

- Оператор присваивания
- Простой и составной операторы
- Условный оператор
- Оператор множественного выбора
- Оператор цикла с предусловием
- Оператор цикла с постусловием
- Оператор цикла с параметром
- Оператор и процедуры безусловного перехода

ОПЕРАТОР ПРИСВАИВАНИЯ

Оператором называется конструкция языка программирования, служащая для задания какого-либо действия или последовательности действий над данными в программе. Совокупность операторов программы реализует заложенный в ней алгоритм.

Процесс «засылки» значения в переменную называется **присваиванием** (первое присваивание называется **инициализацией**). Присваивание осуществляется с помощью специальной конструкции – **оператора присваивания**:



```
Var W, H : Integer;  
Begin  
    W := 23;  
    H := 17;  
    W := W * H  
End.
```

ПРОСТОЙ И СОСТАВНОЙ ОПЕРАТОРЫ

Оператор в программе – это единое неделимое предложение, выполняющее какое-либо действие. **Простой оператор** не содержит в себе других операторов (оператор присваивания, оператор безусловного перехода,...).

Два последовательных оператора должны разделяться точкой с запятой (имеет смысл конца оператора):

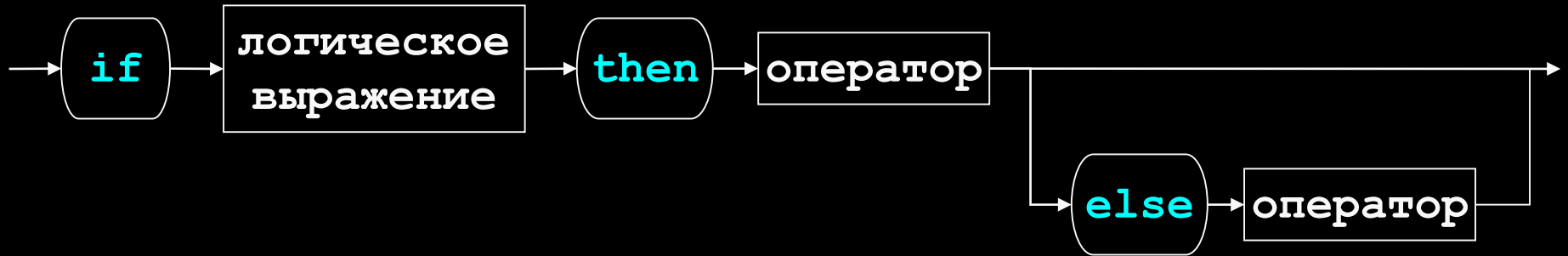
```
a := 11;  b := a * a;  Write(a,b) ;
```

Составной оператор – это последовательность операторов, рассматриваемых как единый. Оформляется с помощью зарезервированных слов begin и end (операторные скобки).

```
begin
    a := 11;
    b := a * a;
    Write(a,b)
end;
```

УСЛОВНЫЙ ОПЕРАТОР

Условный оператор используется для программирования ветвлений, т.е. ситуаций, когда возникает необходимость при определенных условиях выполнять различные действия. Условный оператор имеет структуру:



В каждой ветви допускается запись только одного оператора.

```
if K > 5 then
  begin
    X := X + 5; Y := 1
  end
else
  Y := -1;
```

ВЛОЖЕННЫЕ УСЛОВНЫЕ ОПЕРАТОРЫ

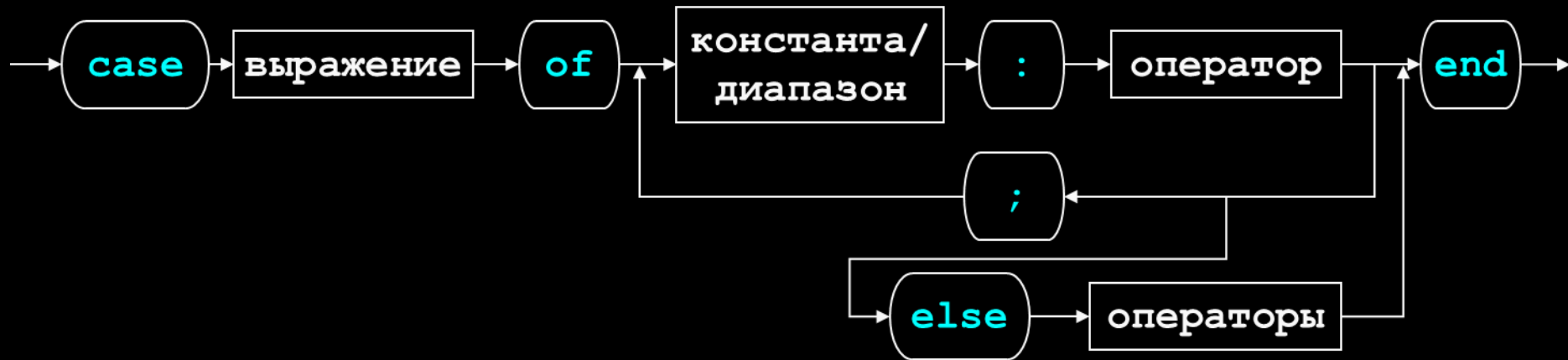
Альтернатива else считается принадлежащей ближайшему условному оператору if, не имеющему своей ветви else.

```
if <условие 1> then
    if <условие 2> then
        <оператор А>
    else
        <оператор Б>;
```

```
if <условие 1> then
    begin
        if <условие 2> then
            <оператор А>
        end
    else
        <оператор Б>;
```

ОПЕРАТОР МНОЖЕСТВЕННОГО ВЫБОРА

Оператор выбора используется для реализации нескольких альтернативных вариантов действий, каждый из которых соответствует своим значениям некоторого параметра.



Значение <выражения>, а также <константа/диапазон> должны относиться к одному из порядковых типов.

В зависимости от значения <выражения> выбирается тот оператор, которому предшествует константа выбора, равная вычисленному значению (альтернатива – операторы после else).

Значения констант должны быть уникальными в каждом наборе, т.е. они могут появиться только в одном варианте.

ОПЕРАТОР МНОЖЕСТВЕННОГО ВЫБОРА

```
case I of                                {I : Word}
    1      :  X := X +1;
    2,3    :  X := X +2;
    4..9   :  begin
                Write(X) ;
                X := X + 3    {м.б. ";" }
            end              {м.б. ";" }
    else
        X := X * X;
        Writeln(X)          {м.б. ";" }
end;
```

ОПЕРАТОР ЦИКЛА "ПОКА" (С ПРЕДУСЛОВИЕМ)

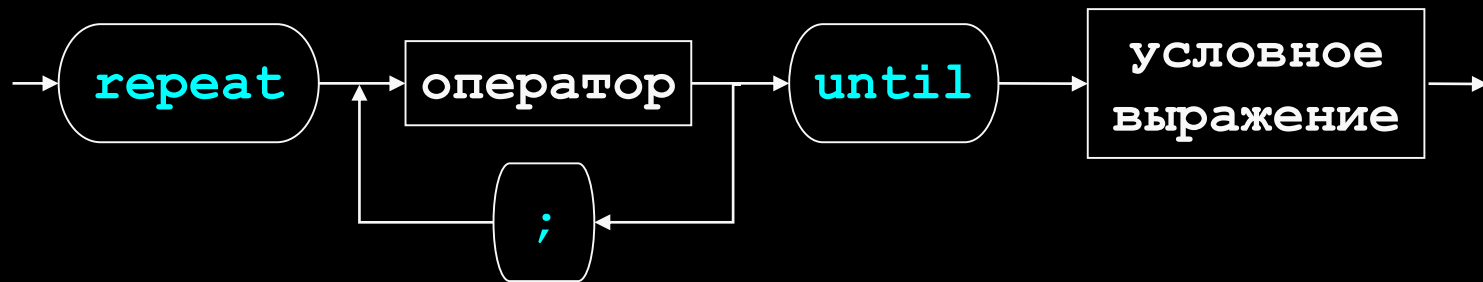


<Оператор> (**тело цикла**), стоящий после служебного слова do, будет выполняться циклически до тех пор, пока выполняется логическое условие, т.е. пока значение <условного выражения> равно True.

Чтобы цикл имел шанс когда-либо завершиться, содержимое его тела должно влиять на условие цикла. Условие должно состоять из корректных выражений и значений, определенных еще до первого выполнения тела цикла.

```
Var    F,N : Int64;                {вычисление 10!}  
Begin  
    F := 1; N := 1;  
    while N <= 10 do  
        begin  
            F := F * N; Inc(N)      {N := N + 1}  
        end;  
    Writeln(F)  
End.
```


ОПЕРАТОР ЦИКЛА "ДО" (С ПОСТУСЛОВИЕМ)



Операторы между словами repeat и until образуют **тело** цикла.

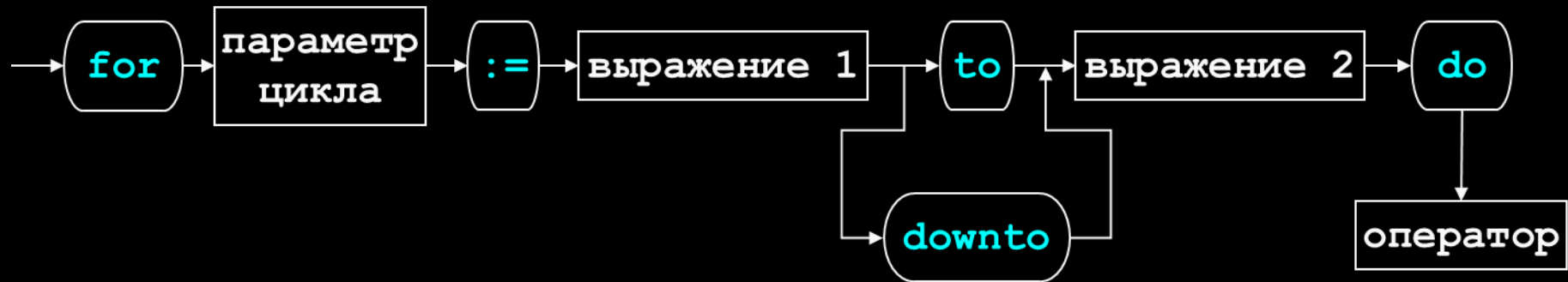
Если <условное выражение> имеет значение True, то цикл завершается.

Цикл "Пока" – "Пока условие истинно, выполнять операторы тела".

Цикл "До" – "Выполнять тело цикла до тех пор, пока не станет истинным условие";

ОПЕРАТОР ЦИКЛА С ПАРАМЕТРОМ (ЦИКЛ ПО СЧЕТЧИКУ)

Используется для организации "строгих" циклов, которые должны быть проделаны заданное число раз.



<Параметр цикла> – переменная порядкового типа, к этому же типу должны относиться значения <выражения 1> и <выражения 2>.

Значение <параметра цикла> меняется в возрастающем (при использовании зарезервированного слова to) или убывающем (downto) порядке от значения <выражения 1> до значения <выражения 2> с постоянным шагом, равным интервалу между двумя ближайшими значениями в типе, к которому относится <параметр цикла> (для целочисленных типов – это 1, для символьного – от одного символа к другому при увеличении кода на 1, и т.д.).

Циклы for допускают вложенность, если никакой из вложенный циклов не использует и не модифицирует переменные – параметры внешних циклов.

ОПЕРАТОР ЦИКЛА С ПАРАМЕТРОМ (ЦИКЛ ПО СЧЕТЧИКУ)

Var

```
I : Integer;  
C : Char;  
B : Boolean;  
E : (E1,E2,E3,E4) ;
```

Begin

```
for I := -10 to 10 do Write(I) ;  
for C := 'a' to 'z' do Write(C) ;  
for B := False to True do Write(B) ;  
for E := E1 to E4 do  
    begin                                {составной оператор}  
        I := ORD(E) ;  
        Write(I)  
    end
```

End.

ОПЕРАТОР БЕЗУСЛОВНОГО ПЕРЕХОДА

Оператор безусловного перехода передает управление выполнением в указанное с помощью метки место программы (является "лишним" с точки зрения теории структурного программирования).

Синтаксические диаграммы
<объявления меток> и
<оператора безусловного
перехода>



Метка может стоять в программе в любом месте между операторами и отделяется от второго оператора двоеточием ":".

Область действия операторов безусловного перехода строго локализована. Запрещены переходы по оператору `goto` между процедурами, а также между основным блоком и процедурой.

```
Label L1, L2;  
Begin  
...  
    goto L1;  
...  
L1 : goto L2;  
...  
L2 : End.
```

ПРОЦЕДУРЫ БЕЗУСЛОВНОГО ПЕРЕХОДА

Процедуры Exit и Halt специально предназначены для выхода из программных блоков (процедур, функций, основного программного блока).

Halt (<код завершения>) осуществляет выход из программы, возвращая операционной системе заданный код завершения.

Exit осуществляет безусловный выход из подпрограммы. Если процедура использована в основном блоке, то она выполняется аналогично Halt.

Процедуры неструктурной передачи управления при работе с циклическими структурами:

Break – реализует выход из цикла любого типа;

Continue – осуществляет переход на следующую итерацию цикла, игнорируя оставшиеся до конца тела цикла операторы.