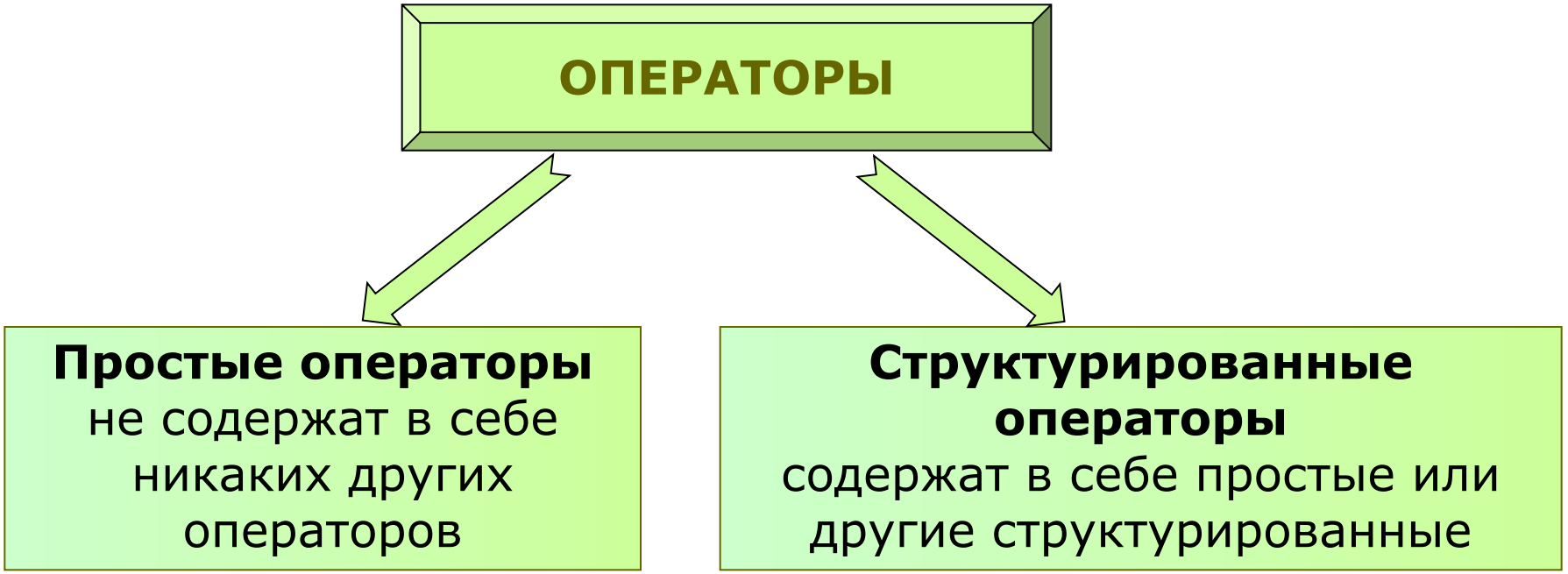
ОПЕРАТОРЫ

Язык программирования
DELPHI

Общие положения

Основная часть программы - последовательность операторов, выполняющих некоторое действие над данными.

Операторы выполняются последовательно и отделяются один от другого точкой с запятой.



Оператор присваивания

Оператор присваивания (**:=**) вычисляет выражение, заданное в его правой части, и присваивает результат переменной, записанной в левой части.

ПР: X := 4; Y := 6; Z := (X + Y) / 2;

Тип выражения должен быть совместим с типом переменной.

Выражение с более узким диапазоном возможных значений можно присвоить переменной с более широким диапазоном значений.

Оператор присваивания

ПР:

```
var  B: Byte;
     K: Integer;
     R: Real;
begin
    B := 255;
    K := B + 1;    // K = 256
    R := K + 0.1; // R = 256.1
    K := R;        // Ошибка!
end.
```

Оператор вызова процедуры Пустой оператор

Оператор вызова процедуры - имя процедуры (стандартной или пользовательской), после которого в скобках перечисляются фактические параметры, передаваемые в процедуру. Более подробно рассмотрим позже.

Примеры вызова стандартных процедур:

```
Inc(x,3);
ShowMessage(''); // Вызов процедуры
```

Пустой оператор

```
;
```

Оператор безусловного перехода

Если после выполнения некоторого оператора надо выполнить не следующий по порядку, а какой-либо другой, отмеченный **меткой**, оператор, используют **оператор безусловного перехода**.

Метка - именованная точка в программе, в которую можно передать управление.

В разделе операторов **метка записывается с двоеточием**.

Раздел описания меток начинается словом **Label**.

Пример описания меток: Label Label1, Label2;

Оператор безусловного перехода

Переход на метку выполняется с помощью **goto**, за которым следует имя метки.

```
ПР:  . . . . . . . . . . . . . .
     label  M1, M2;
     . . . . . . . . . . . . . .
     begin
      M1: Write('Желаем успеха ');
          goto M2;
          Write('Этого сообщения вы не
увидите!');
      M2: Writeln('в освоении среды Delphi!');
          Writeln('Press Enter to exit...');  Readln;
     end.
```

Составной оператор

Составной оператор - группа из произвольного числа операторов, отделенных друг от друга точкой с запятой и заключенную в операторные скобки — **begin** и **end**

```
Begin
    <оператор 1>;
    <оператор 2>;
     …
    <оператор N>
End;
```

Широко используется с **условными операторами** и **операторами повтора**.

Оператор ветвления if (полная форма)

Формат описания:

if <условие> **then** <оператор 1> **else** <оператор 2>;

Внимание! Перед словом **else** точка с запятой **не** ставится.

Условие - выражение булевского типа (может быть простым или сложным). **Сложные условия** образуются с помощью логических операций и операций отношения.

Оператор ветвления if (полная форма)

Логика работы:

Если условие истинно, выполняется оператор 1, если же условие ложно, выполняется оператор 2.

ПР: if K<>0 then Result:=5/K else Result := 0;

if (K<0) or (k>100) then goto M1;

Внимание! Если в какой-нибудь ветви нужно выполнить несколько операторов, то используйте **составной оператор**.

Оператор ветвления if (неполная форма)

Формат описания:

if <условие> **then** <оператор 1>;

Логика работы:

Если условие истинно, выполняется
оператор 1, если же условие ложно,
оператор 1 не выполняется.

ПР: if K<>0 then Result:=5/K;

Оператор ветвления if (неполная форма)

Один оператор **if** может входить в состав другого оператора **if**.

В этом случае говорят о вложенности операторов.

При вложенности операторов каждое **else** соответствует тому **then**, которое непосредственно ему предшествует.

ПР:
```
if K>0 then Result:=1
        else if K=0 then Result:=0
                else Result:=-1;
```

Оператор выбора CASE

Если необходимо сделать **выбор** из конечного числа имеющихся **вариантов**, используем оператор **case**.

Формат описания:

```
case < переключатель> of
   <Значение 1> : <оператор 1>;
    ...
   <Значение n> : <оператор n>;
   else <оператор>;
end;
```

Else может отсутствовать

Оператор выбора CASE

Переключатель – выражение порядкового типа

Значение 1,…,Значение n - допустимые значения переключателя

Оператор 1,…,Оператор n – операторы (могут быть составными)

Логика работы:

Оператор **case** вычисляет значение **переключателя**, затем последовательно просматривает **список его допустимых значений** в поисках вычисленного значения.

Если это значение найдено, выполняет соответствующий ему **оператор**.

Если переключатель не попадает ни в один из списков, выполняется оператор, стоящий за словом **else**.

Оператор выбора CASE

Список значений переключателя может содержать **константы** и **диапазоны**, отделенные друг от друга запятыми.

Границы диапазона записываются двумя константами через разграничитель в виде двух точек. ПР: 20..31

Внимание! Тип значений должен быть совместим с типом переключателя.

Оператор выбора CASE

ПР:

```
case Day of
  20..31: Writeln('День в диапазоне 20 - 31.');
  1, 5..10: Writeln('День в диапазоне 1, 5 - 10.');
  else Writeln('День не попадает в заданные
   диапазоны.');
end;
```

Внимание! Если значения переключателя записаны в возрастающем порядке, то поиск требуемого оператора выполняется значительно быстрее.

Операторы повтора - циклы

Алгоритм решения многих задач требует **многократного повторения** одних и тех же действий. При этом суть действий остается прежней, но меняются данные.

Для многократного (циклического) выполнения одних и тех же действий предназначены **операторы повтора (циклы)**.

К ним относятся операторы **for**, **while** и **repeat**.

Оператор повтора FOR…TO

Оператор повтора **for** используется, если **заранее известно количество повторений цикла**.

for <параметр цикла> **:=**<начальное значение> **to** <конечное значение> **do** <оператор>;

Параметр цикла - переменная порядкового типа данных

Начальное и конечное значения - выражения, определяющие начальное и конечное значения параметра цикла (они вычисляются только один раз перед началом работы цикла)

Оператор образует тело цикла (может быть составным).

Оператор повтора FOR…TO

Логика работы:

Оператор **for** обеспечивает выполнение тела цикла до тех пор, пока не будут перебраны все значения параметра цикла от начального до конечного.

Если начальное значение параметра цикла **больше** конечного значения, цикл не выполнится ни разу.

После каждого повтора значение параметра цикла **увеличивается** на единицу.

ПР:
```
for K:=1 to 10 do
  ShowMessage(IntToStr(K));
      // На экран выводятся целые числа от 1 до
```

Оператор повтора FOR..DOWNTO

Формат описания:

for <параметр> **:=**<начальное значение> **downto** <конечное значение> **do** <оператор>;

Логика работы:

Оператор **for** обеспечивает выполнение тела цикла до тех пор, пока не будут перебраны все значения параметра цикла от начального до конечного.

Если начальное значение параметра цикла **меньше** конечного значения, цикл не выполнится ни разу.

После каждого повтора значение параметра цикла **уменьшается** на единицу.

Оператор повтора FOR..DOWNTO

ПР:

```
for K:=10 downto 1 do
    ShowMessage(IntToStr(K));
  // На экран выводятся
последовательно
  // целые числа от 10 до 1
  // в порядке убывания.
```

Оператор повтора REPEAT

Оператор **repeat** (**цикл с постусловием**) используют, когда тело цикла должно быть выполнено **перед** тем, как произойдет проверка **условия завершения цикла**.

Формат описания:

Repeat

<оператор 1>; ... <оператор N>; // Тело цикла

until <условие завершения цикла>;

Логика работы:

Тело цикла выполняется до тех пор, пока **условие завершения цикла** (выражение булевского типа) не станет истинным.

Оператор повтора REPEAT

Внимание! Между словами **repeat** и **until** может находиться произвольное число операторов без операторных скобок **begin** и **end**

Внимание! Цикл **repeat** выполняется хотя бы один раз

ПР:
```
Randomize;
S := 0;
repeat
    X:= -3+Random(7);  S:=S+X
until X = 0;
// S – сумма сгенерированных чисел X
// Суммирование прекращается, когда X = 0
```

Оператор повтора WHILE

Оператор **while** (**цикл с предусловием**) используют, когда тело цикла должно быть выполнено **после** проверки **условия выполнения цикла**.

Формат описания:

While <условие выполнения цикла> **do**
<оператор>; // Тело цикла

Логика работы:

Перед каждым выполнением тела цикла происходит проверка условия.

Если оно истинно, цикл выполняется и условие вычисляется заново.

Если оно ложно, происходит выход из цикла.

Оператор повтора WHILE

Внимание! Если первоначально условие ложно, то тело цикла не выполняется ни разу.

Внимание! Если тело цикла должно содержать несколько операторов, для представления его как одного составного оператора используйте операторные скобки **begin** и **end**.

ПР:
```
Randomize; S := 0;
while X<>0 do
begin
    X:= -3+Random(7);  S:=S+X
end;
// S – сумма сгенерированных чисел X.
// Суммирование прекращается, когда X = 0.
```

Оператор Continue

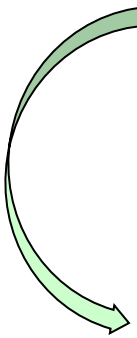
Для управления работой операторов повтора можно использовать процедуру-оператор **Continue**.

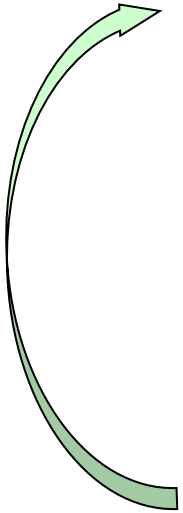
Внимание! Оператор **Continue** можно вызывать только в теле цикла

Continue немедленно передает управление **оператору проверки условия**, пропуская оставшуюся часть цикла

Оператор Continue

```
Repeat
    <оператор 1>;
     ...
    Continue;
     ...
    <оператор N>
until <условие>;
```





```
While <условие>
do
begin
    <оператор 1>;
     ...
    Continue;
     ...
    <оператор N>
end;
```

Оператор Break

Для управления работой операторов повтора можно использовать процедуру-оператор **Break**.

Внимание! Оператор **Break** можно вызывать только в теле цикла

Break прерывает выполнение цикла и передает управление **первому оператору, расположенному за блоком цикла**

Оператор Break

```
Repeat
    <оператор 1>;
     ...
    Break;
     ...
    <оператор N>
until <условие>;

<оператор>;
```

```
While <условие>
do
begin
    <оператор 1>;
     ...
     Break;
     ...
    <оператор N>
end;

<оператор>;
```

Пример использования Continue

```
Randomize;
P:=1;
For K:=1 to 10 do
begin
     X:= -3+Random(7);
      if X=0 then Continue else P:=P*X;
end;
// Будут сгенерированы 10 чисел
// P - произведение ненулевых чисел
```

Пример использования Break


```
Randomize;
P:=1;
For K:=1 to 10 do
begin
     X:= -3+Random(7);
      if X=0 then Break else
   P:=P*X;
end;
// Будет произведена попытка
// сгенерировать 10 чисел.
   Как
// только сгенерируется число
// X=0, произойдет выход
// из цикла
```