Циклы и массивы

Функции цикла в программе

Алгоритмы решения многих задач являются циклическими, т. е. для достижения результата определенная последовательность действии должна быть выполнена несколько раз.

Например, чтобы найти фамилию человека в списке, надо проверить первую фамилию списка, затем вторую, третью и т. д. до тех пор, пока не будет, найдена нужная или не будет достигнут конец списка. Такие повторяющиеся действия называются *циклами*.

Цикл – единственная конструкция, позволяющая возвращаться «назад» по тексту программы.

Виды циклов:

- ✔ *цикл с предусловием;*
- ✔ *цикл с постусловием;*
- ✔ *цикл с параметром.*

Цикл с предусловием

Условие продолжения цикла проверяется до его начала. Цикл выполняется, пока условие истинно. Как только условие становится ложным, цикл заканчивается.

Если условие ложно с самого начала, цикл не выполняется ни разу

Оператор *while* используется в том случае, если некоторую последовательность действий надо выполнить несколько раз, причем необходимое число повторений во время разработки программы неизвестно.
В общем виде инструкция *while* записывается так:

while *условие* **do**
тело цикла;

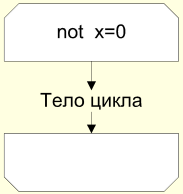
Цикл с предусловием

Пример использования

```
...
x:=10;
while not(x=0) do
 x:=x-1;
```

Запись с отступами

Обозначение на блок схемах



Если перед циклом x=0, тело цикло не будет выполнено ни разу

Телом цикла while может быть составной оператор **begin..end**

Цикл с постусловием

Цикл с постусловием используется, если надо провести некоторые повторяющиеся вычисления (цикл), число повторов во время разработки программы неизвестно, но хотя бы один раз цикл должен выполниться.

В цикле с постусловием задается условие прекращения цикла. Пока оно ложно – цикл продолжается.

В общем виде инструкция *repeat* записывается так:

repeat {инструкции} **until** *условие,*

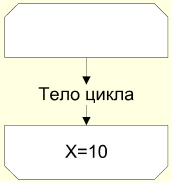
Условие – выражение логического типа, определяющее условие завершения цикла.

Цикл с постусловием

Пример использования

```
Var x:string;
begin
 repeat
  InputQuery('','Введите x:',x)
 until x<>''
…
```

Обозначение на блок схемах



В цикле REPEAT..UNTIL в теле можно писать много операторов без BEGIN..END

Цикл с параметром

Цикл с параметром используется, если некоторую последовательность действий надо выполнить несколько раз, причем число повторений заранее известно. Параметром может быть переменная любого перечислимого типа (целое число, символ, булевское значение).

В общем виде инструкция for записывается так:

+1

-1

for *переменная:=нач_значение* **to** *кон_значение* **do** оператор;
for *переменная:=нач_значение* **downto** *кон_значение* **do** оператор;

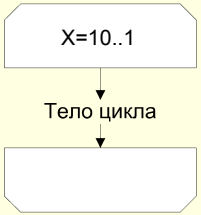
нач_значение – выражение, определяющее начальное значение переменной – счетчика циклов;
кон_значение – выражение, определяющее конечное значение переменной – счетчика циклов.

Цикл с параметром

Примеры использования

```
…
Var x:byte; z:char;
…
For x:=10 downto 1 do
 s:=s+x;
…
For z:='A' to 'Z' do
 c:=c+z;
```

Обозначение на блок схемах



В случае **For x:=10 to 1 do** цикл не выполнится ни разу

В случае **For x:=1 downto 10 do** цикл не выполнится ни разу

Шаг цикла с параметром

Параметр цикла при каждой итерации всегда изменяется на +1 или -1.

Если надо менять переменную с другим шагом, следует делать пересчет

Пример: Вычислить сумму чисел от 1 до 10 с шагом 0.1 (1+1.1+1.2+…+9.9+10)

```
VAR s:longint; c:word;
BEGIN
 s:=0;
 FOR c:=1 to 100 DO
  s:=s+c/10;
```

Нельзя менять значение параметра в теле цикла

После завершения цикла значение его параметра неопределенное

Цикл по четным и нечетным значениям

Пример: просуммировать все четные числа от 1 до 1000

```
VAR s:longint; c:word;
BEGIN
 s:=0;
 FOR c:=1 to 1000 DO
  IF c MOD 2=0 THEN
   s:=s+c;
```

Остаток от деления (MOD) на 2 будет равен нулю для четных чисел и единице для нечетных

Команды BREAK и CONTINUE

Для немедленного выхода из любого цикла можно использовать команду *Break*

<u>Пример использования</u> Найти позицию первого пробела в строке

```
N:=0;
for i:=1 to Length(S) do
if S[i] = ' ' then
 begin
  N:=i;
  Break
 end;
```

Команды BREAK и CONTINUE

Команда *Continue* по своему действию противоположна команде прерывания цикла. Она позволяет немедленно начать новую итерацию, пропустив все оставшиеся после нее операторы в теле цикла.

Пример использования

```
N:=0;
for i:=1 to Length(S) do
Begin
If S[i] <>' ' then
  Continue;
 N:=i;
 Break
End;
```

Вечные циклы

Если в теле цикла не изменяется условие цикла, то получаем вечный цикл, что приводит к зависанию всего приложения.

Пример (не пробовать!!!!)

```
t:=true;
while t do
 x:=100;
```

Зависшая программа снимается нажатием Ctrl+F2 в Delphi

Даже если цикл не вечный, а просто долгий, программа перестает реагировать на действия пользователя и не перерисовывает экран

Реакция на действия пользователя в цикле

По умолчанию любой цикл «запирает» все элементы интерфейса, поэтому не удастся поставить кнопку «Прервать цикл». Чтобы программа в цикле отзывалась на действия пользователя, в теле цикла пишем:

Application.ProcessMessages;

При нажатии на кнопку Stop значение переменной Br устанавливается в True

```
Br:=false;
FOR i:=1 to 1000000000 DO
 BEGIN
   …
   Application.ProcessMessages;
   IF br THEN
    Break
 END;
```

Что такое INC и DEC?

Процедура INC(X) увеличивает значение перечислимого типа на 1 и соответствует выражению X:=X+1

Процедура INC(X,N) увеличивает значение перечислимого типа на N и соответствует выражению X:=X+N

Процедура DEC(X) уменьшает значение перечислимого типа на 1 и соответствует выражению X:=X-1

Процедура DEC(X,N) уменьшает значение перечислимого типа на N и соответствует выражению X:=X-N

Вложенные циклы (nested loops)

Если цикл включает в себя один или несколько циклов, то содержащий внутри себя другие циклы называется *внешним*, а цикл, содержащийся в другом цикле - *вложенным*.

При программировании вложенных циклов необходимо выполнить правило: внутренний оператор цикла и принадлежащая ему область действия должны полностью содержаться внутри области внешнего цикла, таким образом внешний цикл всегда начинается раньше, а заканчивается позже, чем внутренний.

Delphi выдерживает до 255 вложенных циклов

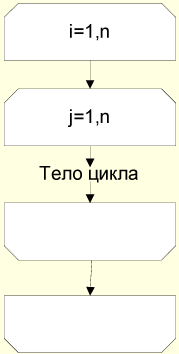
Вложенные циклы

Примеры использования

```
for i :=1 to n do
 for j :=1 to n do
```

```
WHILE x<10 DO
 FOR i:=1 TO 20 DO
  s:=x+i;
```

Блок – схема, соответствующая вложенному циклу



Массивы (arrays)

Массив — это структура данных, представляющая собой набор переменных одинакового типа, имеющих общее имя. Массивы удобно использовать для хранения однородной по своей природе информации, например, таблиц и списков.

Объявление массива

TYPE TA=**array** [нижний_индекс **..** верхний_индекс] **of** тип;
VAR c:TA;
где:
TA— имя типа массива (придумываем сами);
тип — тип данных каждого элемента массива.

Примеры объявления массивов:
koef:**array**[0. .2] **of** integer;
name:**array**[1..30] **of string**[25];

При объявлении массива удобно использовать константы.

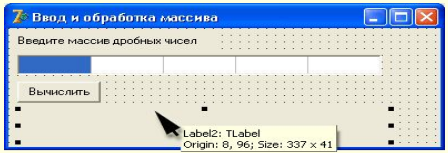
```
const
NT = 18; // число команд
SN = 25; // предельная длина названия команды
TYPE team: array[1..NT] of string[SN];
```

Массивы могут быть константными:
Имя:array [нижний_индекс..верхний_индекс] of тип = (список);
где список — разделенные запятыми значения элементов массива.

Например:
a: array[1..10] of integer = (1,2,3,4,5,6,7,8,9,10);

Ввод-вывод массива

```
TYPE Ta : array[1..5]  of integer; // тип массива
summ: integer; // сумма элементов
a: TA; // массив
sr: real; // среднее арифметическое
i: integer; // индекс
begin
// ввод массива
// считаем, что если ячейка пустая, то
соответствующий
// ей элемент массива равен нулю
for i:= 1 to 5 do
if Length(StringGridl.Cells[i-1, 0]) <>0
then a[i] := StrToInt(StringGridl.Cells[i-1,0])
else a[i] := 0;
// обработка массива
summ := 0;
for i :=1 to 5 do
summ := summ + a[i]; sr := summ / 5;
//вывод результата
Label2.Caption :='Сумма элементов: ' +
IntToStr(summ)
+ #13+ 'Среднее арифметическое: ' + FloatToStr(sr);
end;
```



Поиск минимального (максимального) элемента массива

```
const
SIZE=5; TYPE TA=array[1..SIZE] of integer;
var
a:TA; // массив целых
min:integer; // номер минимального элемента массива
i:integer; // номер элемента, сравниваемого с минимальным
begin
// ввод массива
for i:=1 to SIZE do
 a[i]:=StrToInt(StringGridl.Cells[i-1,0]);
// поиск минимального элемента
min:=a[1]; // пусть первый элемент минимальный
for i:=2 to SIZE do
if a[i]< a[min] then min:=i;
// вывод результата
label2.caption:='Минимальный элемент массива:'
+IntToStr(a[min] +#13+'Номер элемента:'+ IntToStr(min);
end;
```

Многомерные массивы

В общем виде инструкция объявления двумерного массива выглядит так:

array[НижняяГраница1..ВерхняяГраница1,
НижняяГраница2..ВерхняяГраница2] **of** Тип

Никаких строк и столбцов в таком массиве нет!

Вложенные циклы:

```
FOR i:=1 TO m1 DO
 FOR j:=2 TO m2 DO
  a[I,j]:=0;
```