Операторы цикла

Оператор цикла

- В языке С операторы цикла служат для многократного выполнения последовательности операторов до тех пор, пока выполняется некоторое условие
- Условие может быть установленным заранее (как в операторе for) или меняться при выполнении тела цикла (как в while или do-while)

Цикл for

Общая форма оператора for:

for (*инициализация*; *условие*; *приращение*)

оператор;

Инициализация – оператор присваивания, который задает начальное значение счетчика цикла

Условие – условное выражение. Пока оно истинно цикл выполняется

Приращение – изменяет значение счетчика цикла при очередном его выполнении

```
for (инициализация; условие; приращение)
    {
        оператор 1;
        оператор 2;
        …
        оператор n;
    }
```

Пример 1

```
#include <stdio.h>
 int main(void)
{
   int x;
   for(x=1; x <= 100; x++) printf("%d ", x);
   return 0;
}
 for(x=100; x != 65; x -= 5)
{
   z = x*x;
   printf("Квадрат %d равен %d", x, z);
}
```

Пример 2

```
x = 10;
for(y=10; y!=x; ++y)
printf("%d", y);
printf("%d", y); /* Это единственный printf() который
                    будет выполнен */
```

Пример 3

```
 int main(void)
{
   int Summa=0;
   for(int i=1; i <= 10; i++)
     Summa=Summa+i;
   printf("%d ", Summa);
   return 0;
}
```

Summa=0

НЦ

i=1: Summa=0+1=1;

Пример 4

```
 int main(void)
{
   int Summa=0;
   for(int i=1; i <= 100; i++)
   {
   if (i%2==0) Summa=Summa+1;
   printf("%d ", Summa);
   }
   return 0;
}
```

Варианты цикла for

Использование нескольких счетчиков цикла – наиболее распространенный вариант.

Например, в данном примере используется два счетчика одновременно:

```
for(int i=0, j=0; i+j<=10; i++)
{
   j=getchar();
   …
}
```

Операторы инициализации счетчиков разделены запятой, то есть они выполняются последовательно. При каждой итерации значение переменной i увеличивается на 1, а переменная j вводится с клавиатуры.

Варианты цикла for

Пропуск разделов цикла. Так как каждый из разделов цикла является необязательным, то их можно пропускать

Данный цикл выполняется до тех пор, пока пользователь не введет число 100:

```
for(i=1; i!=100; )scanf("%d ", &i);
```

Бесконечный цикл. Так как все разделы являются необязательными, цикл for легко можно сделать бесконечным, не задав условного выражения:

```
for( ; ; ) printf("бесконечный цикл");
```

Пустой цикл. Тело цикла может не содержать ни одного оператора, такой цикл называется пустым. Это можно использовать для повышения эффективности некоторых алгоритмов и задержки выполнения программы

Например, одна из распространенных задач – удаление пробелов из входного потока. Эту задачу решает цикл for, который пропускает все пробелы, стоящие перед словами в строке str:

```
for( ; *str== ' '; str++);
```

Цикл с предусловием – оператор цикла while

Цикл while с предусловием в общем виде записывается:

```
while(условие)оператор;
```

Цикл выполняется пока значение условия истинно. Условие считается истинным, если значение выражения не равно 0.

Например,

```
int i =0;
while(i<10){
   printf("%d", i);
   i=i+1;
}
```

Цикл выполняется пока значение переменной i не равно 10

Оператор безусловного перехода break

```
int i =0, j=20;
while(i<10)
{
   printf(“%d”, i);
   if(j==10)break;
   i = i + 1;
  j = j – 1;
}
```

Цикл завершится, если значение j станет равным 10

Оператор безусловного перехода continue

оператор continue прерывает текущую итерацию и начинает новую, в противном случае выполняются операторы, стоящие после ключевого слова continue

```
gets(s);
str=s;
space=0;
while(*str){
   if(*str != ‘ ‘)continue;
   str++;
   space++;
}
```

Цикл с постусловием – оператор цикла do-while

Если требуется выполнить тело цикла, хотя бы один раз до проверки условия выхода, используют оператор do-while:

```
do{
    оператор;
} while(условие);
```

Цикл выполняется пока значение условия истинно. Например, считываем и выводим на экран число, пока не будет введено число большее или равное 100:

```
int i;
do{
    scanf("%d ", &i);
    printf("%d", i);
} while(i<100);
```

В этом случае появляется возможность просмотра значения переменной i сразу же после ее инициализации вне зависимости, как будет выполняться цикл.

Пример использования do-while

```
void menu(void)
{
char ch;
printf("1. Проверка правописания\n");
printf("2. Коррекция ошибок\n");
printf("3. Вывод ошибок\n");
printf(" Введите Ваш выбор: ");
do
{
   ch = getchar(); /* чтение выбора с клавиатуры */
   switch(ch)
   {
   case '1': check_spelling(); break;
   case '2': correct_errors(); break;
   case '3': display_errors(); break;
```

Вложенные циклы

Очень часто в программе требуется организовать выполнение двух или более вложенных циклов. Главное правило, которое нужно помнить – не допускать повторяющихся названий переменных цикла.

К примеру, имеем:

```
for(int i=0; i < 10; i++)
 {
   for(int j=0; j<10; j++)
     printf("%d", j+i*10);
     printf("\n");
}
```

Результатом выполнения этой программы будет матрица, заполненная элементами в порядке возрастания их значений.