ЕН.Ф.02 – Информатика и программирование

Лекция 4. Операторы цикла

Конова Елена Александровна
E_Konova@mail.ru

Поток управления в алгоритмах

Программа – линейная последовательность операторов, определяющая поток управления.

Последовательность выполнения операторов программы может изменяться в зависимости от условий, сложившихся при ее выполнении. Изменяют эту последовательность операторы управления.

1. Условный оператор организует ветвление при общей линейной схеме.
2. Оператор переключения (**`switch`**) организует ветвление по нескольким направлениям.
3. Операторы цикла организуют повторение фрагментов алгоритма при общей линейной схеме.

Виды циклов

1. Арифметический (управляемый счетчиком). Как правило, повторяется заранее известное число раз.

Пример: спортсмен должен пробежать 10 кругов.

Пример: найти сумму 15-ти слагаемых.

2. Итерационный (управляемый событием). Как правило, число повторений заранее неизвестно.

Пример: спортсмен должен бежать, пока не устанет.

Пример: найти сумму с указанной точностью.

В любом случае управление циклом выполняет некая величина, которая называется «параметр цикла» или управляющая переменная. Она, как правило, изменяется в теле цикла и позволяет завершить работу цикла.

Этапы выполнения цикла

1. Подготовка цикла: действия, которые не относятся непосредственно к логической схеме цикла, но позволяют правильно выполнить цикл. Выполняются однократно перед входом.
2. Точка входа в цикл: момент передачи управления первому оператору тела цикла.
3. Итерация: очередное выполнение тела цикла.
4. Точка проверки условия: момент проверки условия, при котором решается, делать ли новую итерацию, или перейти к оператору, стоящему за циклом (в зависимости от синтаксиса).
5. Выход из цикла: передача управления оператору, стоящему за циклом.

Составные части цикла

Подготовка включает присваивание стартовых значений переменным, участвующим в управлении или выполнении цикла.

Тело цикла – фрагмент, который должен быть повторен многократно.

Изменение параметра цикла.

Проверка условия завершения цикла.

Роль параметра цикла

1. В процессе подготовки цикла управляющая переменная принимает стартовое значение.
2. В теле цикла, которое повторяется многократно, происходит изменение параметра цикла.
3. В проверке условия завершения цикла параметр цикла присутствует явно или нет.

Вывод: как правило, параметр цикла изменяется в теле цикла и позволяет завершить работу цикла.

Выбор управляющей переменной определяется логикой задачи.

Операторы цикла

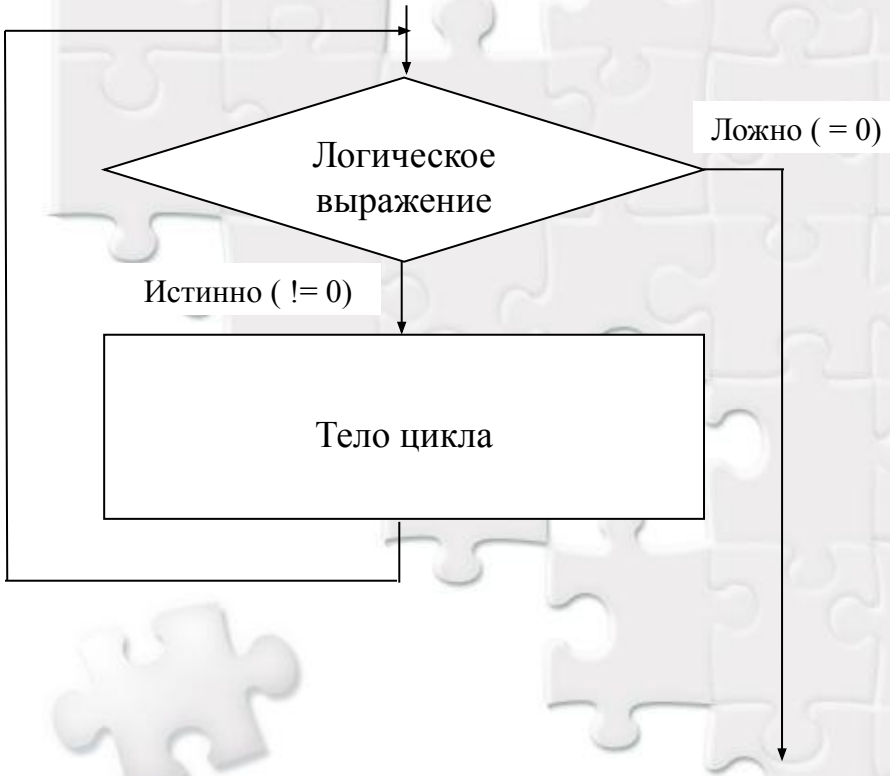
`while` – цикл с предусловием,

`do..while` – цикл с постусловием,

`for` – цикл, управляемый счетчиком.

Назначение: организация многократного повторения произвольного фрагмента программы.

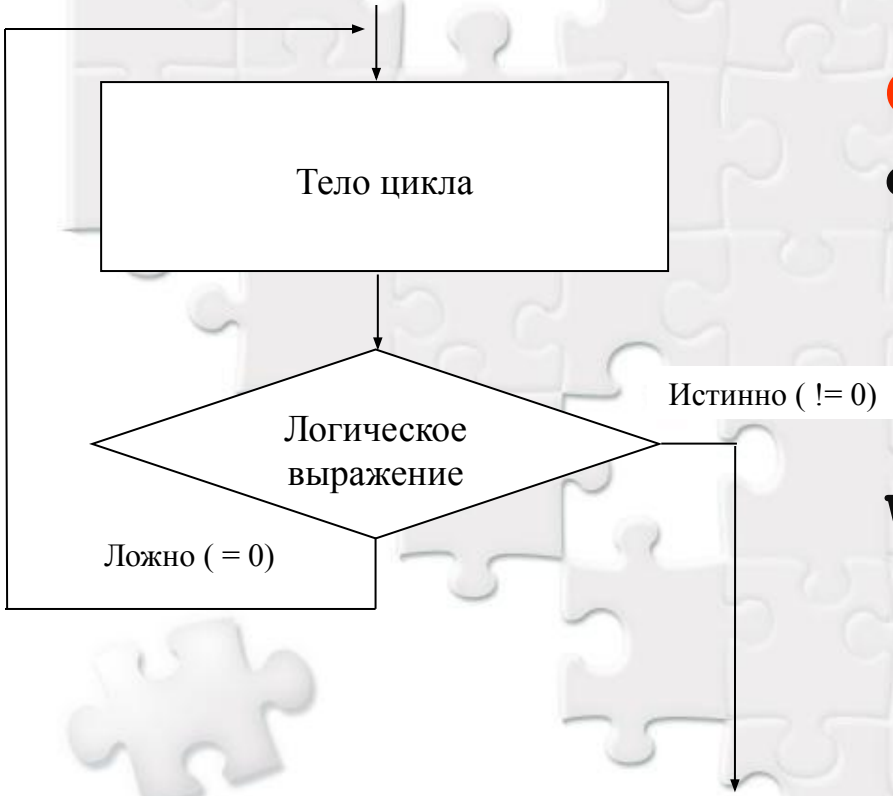
Оператор цикла **while**



Синтаксис:

```
while (Логическое_выражение)
  {
    // Тело цикла;
  }
```

Оператор цикла **do…while**



Синтаксис:

```
do
  {
  // Тело цикла;
   }
while(Логическое_выражение);
```

Составляющие

Тело цикла – один или несколько операторов, в общем случае составной оператор или блок {}.

Логическое_выражение – условие завершения цикла. Является выражением целого типа, значение которого может быть == 0 (Ложно) или !=0 (Истинно).

Семантика **`while`** : тело цикла выполняется всегда, когда Логическое_выражение имеет значение !=0. Когда Логическое_выражение == 0, управление передается оператору, стоящему за циклом.

Семантика **`do`**...**`while`**: тело цикла выполняется многократно, пока Логическое_выражение !=0. Как только выражение == 0, цикл заканчивается, управление передается оператору, стоящему за циклом.

Особенности операторов цикла

Особенности **`while`** : проверка условия происходит до выполнения тела цикла, поэтому для заведомо ложного выражения тело цикла не будет выполнено ни разу.

Особенности **`do`**...**`while`** : проверка условия происходит после выполнения тела цикла, поэтому, как бы ни было задано Логическое_выражение, оператор тела цикла выполнится хотя бы один раз. Использование удобно, когда условие не определено при входе (управление событием).

Схема цикла, управляемого счетчиком

```
// Оператор while
int Count;
Count = 1;
while (Count <= 10)
    {
   // Тело цикла.
   Count ++;
    }
```

```
// Оператор do… while:
int Count;
Count = 1;
do
    {
     // Тело цикла.
   Count ++;
    }
while (Count <= 10)
```

Оператор цикла `for`

Назначение – организация арифметических и итерационных циклов.

Синтаксис: // В заголовке цикла – все управление.

```
for (Выражение1; Выражение2; Выражение3)
  {
     // Тело цикла;
  }
```

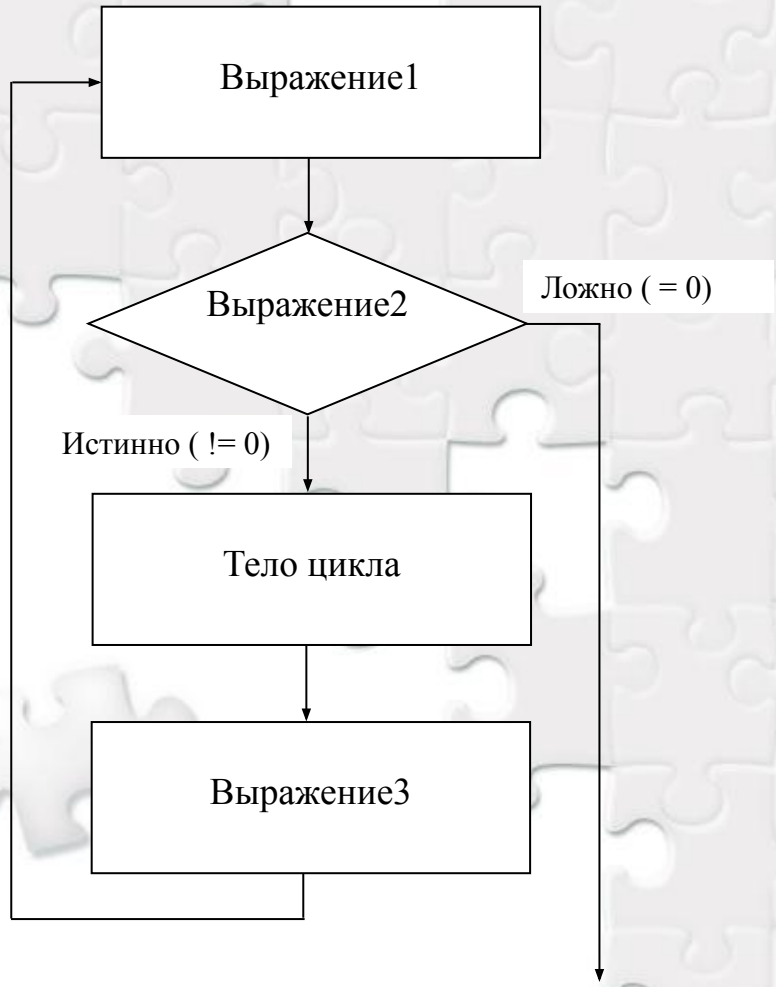
Выражение1 задает начальное значение параметра цикла.

Выражение2 (логическое) задает условие завершения выполнения цикла.

Выражение3 задает приращение параметра цикла.

Тело цикла – как правило, составной оператор {}.

Семантика



Перед входом в цикл однократно выполняется Выражение1.

Тело цикла выполняется многократно, пока Выражение2 (условие) отлично от 0.

В теле цикла выполняется Выражение3.

Замечание 1

Первое и третье выражения (второе тоже, но в этом нет смысла) могут состоять из нескольких выражений, отделенных запятой. Их смысл, это:

а) подготовка цикла,

б) приращение параметра или действие.

Например,

```
for ( S=0, n=1; n<=N; n++)
    S+=n;     // Накопление суммы.
for (i=0, j=N-1; i<N/2; i++, j--)
    a[i] = b[N-j+1]; // Инвертирование.
```

Замечание 2

Некоторые (все) выражения могут отсутствовать, тогда знак «;» не опускается.

```
for (S=0,n=1; n<N; )    // Приращение в теле цикла.
    S += n++;
//--------------------------------
S=0;
n=1;
for ( ; n<N; )
    S += n++;
//--------------------------------
S=0;
n=1;
for ( ; ; )
    {
     S += n++;
     if (n>N) break;
    }
```

Замечание 3

Рабочая переменная может быть объявлена в теле цикла, тогда она известна в пределах охватывающего блока.

```
S=0;
for (int i=0; i<N; i++)
    S+=i;
```

Особенности семантики for

1. Проверка условия происходит до выполнения тела цикла (как **`while`**).
2. Приращение управляющей переменной происходит после выполнения тела цикла.
3. Если условий выхода несколько, то выход происходит по первому условию.
4. Управляющая переменная не обязательно целого типа.

Два вопроса к исследователям

1. О роли операции ++ в приращении цикла.

```
for (int i=1; i<=I; i++)       // ?
    {
        ...
    }
for (int i=1; i<=I; ++i)       // ?
    {
        ...
    }
```

2. Зачем так много операторов цикла?

Операторы **`break`** и **`continue`**

Операторы прерывания **`break`** и продолжения **`continue`** используются для циклов **`do`**, **`while`**, **`for`** и переключателя **`switch`**, изменяя поток управления.

Оператор **break**

Назначение: прекращает выполнение цикла с передачей управления следующему за циклом оператору.

Синтаксис:

```
break;
```

Особенность – значения всех переменных сохраняются, в отличие от операторов цикла, в которых принято считать, что значение управляющей переменной теряется (не определено).

Замечание. В случае вложения циклов **break** прерывает только непосредственно охватывающий цикл.

Пример. Найти сумму арифметической прогрессии, не превышающей указанного значения N.

Оператор `continue`

Назначение: переход к следующей итерации тела цикла без прекращения выполнения.

Синтаксис:

```
continue;
```

Семантика: в любой точке цикла прервет текущую итерацию и перейдет к проверке условия завершения цикла.

Необходимость – обработка исключительных ситуаций в теле цикла.

Пример. Найти сумму слагаемых вида 1/n, исключая ноль в знаменателе.

$$S = \sum_{x=-1}^{x=1} \frac{1}{x}$$

Оператор **switch**

Назначение: Организация разветвления алгоритма на несколько взаимоисключающих ветвей.

Синтаксис:

```
switch (Выражение) {
        case значение1; {     // Ветвь_1
         блок_1
         break;
         }
.....................................
        case значениеN: {     // Ветвь_N
         блок_N
         break;
         }
        default: {          // По умолчанию
         блок_N+1
         break;
           }
  }
```

Оператор **switch**

Выражение, это любое выражение, которое может принять одно из нескольких прогнозируемых значений:

выражение = {значение1,... ЗначениеN};

Тип целочисленный: **char**, все клоны **int**, **enum**

Семантика

1. Вычисляется значение выражения.
2. Ветвление:

Если значение выражения равно значение_i, то выполняется блок_i.

Если значение выражения не равно ни одному из перечисленных значений, то выполняется блок **default**.

Оператор **switch**

Замечание.

Инструкция **break**; прерывает поток управления и передает управление оператору, следующему за **switch** – оператор работает как переключатель.

Если **break**; опущен, то будут выполнены все инструкции, следующие за данной веткой до конца оператора или до встреченного **break**; – оператор работает как выбор.

Замечание.

Ветвь **default** может отсутствовать, если программист уверен, что иных значений выражение принять не может.

Оператор **switch**

Пример: переключение по символьному выражению.

```
// Sign = Знак + или - или * или / (1,2,3,4)
S = 0;
switch (Sign)
{
    case '+' :
     S += x; break;
    case '-' :
     S -= x; break;
    default  :
     S = x;
}
```

Оператор **switch**

Примеры выбора:
Оценка = 5 или 4 = "Зачтено", иначе = "Не зачтено"

```
switch (Grade)  //Оценка = 2,3,4,5
{
 case 5 :
 case 4 :printf("Зачтено\n");     // Если балл 4 или 5
   break;
 case 3:
 case 2: printf("Не зачтено\n"); // Если балл 3 или 4
   break;
 default:
   printf("Ошибка ввода\n");
   break;
}
}
```