Занятие № 3. Циклы

Циклы применяются для того, чтобы C++ мог автоматически выполнять некоторые действия многократно.

Цикл for

Общий синтаксис оператора цикла for имеет вид:

```
for(<инициализация переменных управления циклом>;
    <проверка продолжения цикла>;
<модификация переменных управления циклом, часто
их приращение или уменьшение>)
```

Пример

```
for(i = 0;i < 10;i++)
cout<<"The current count is "<<i<<"."<<endl;
```

Оператор цикла for имеет три компонента, каждый из них необязателен.

Первый компонент **инициализирует переменные управления** циклом. (C++ предоставляет вам возможность использовать более одной переменной управления циклом).

Второй компонент цикла — это **условие**, которое определяет, будет ли цикл выполнять следующую итерацию (нечто вроде оператора if, но без самого ключевого слова if).

Последний компонент цикла for — предложение, которое **изменяет переменные управления циклом**; часто это просто операция инкремента и/или декремента.

Открытые циклы, использующие цикл for

Если вы оставите все три компонента цикла пустыми, результатом будет *открытый цикл* (open loop).

C++ позволяет вам выходить из цикла следующими четырьмя способами:

1. Оператор **break** вызывает переход к выполнению кода, следующего за текущим циклом, во многом подобно тому, как он мог бы быть использован, чтобы продолжить выполнение вне оператора **switch**. Используйте оператор **break**, если вы хотите выйти из цикла и продолжить работу оставшейся части программы.

C++ позволяет вам выходить из цикла следующими четырьмя способами:

2. Оператор **return** осуществляет возврат из текущей функции (включая main).

3. Оператор **throw** вызывает генерацию исключения. Это используется, если произошла ошибка и вы не можете продолжать выполнение оставшейся части программы без какого-либо обработчика ошибок. Однако применяйте этот метод с осторожностью; исключения, строго говоря, предназначены для использования в нештатных обстоятельствах — в случае ошибок, например.

4. В самых чрезвычайных случаях выйти из программы можно с помощью функции **exit** (объявленной в заголовочном файле STDLIB.H). Функция exit прекратит выполнение итераций и при ведет к выходу из программы.

Цикл do-while

Цикл do-while в C++ — это условный цикл, который проверяет итерационное условие в конце цикла. Поэтому в цикле do-while всегда выполняется по крайней мере одна итерация.
Условный цикл выполняется до тех пор, пока условие имеет значение true.

Цикл do-while имеет следующий общий синтаксис:

```
do {
   < последовательность операторов>
} while (условие);
```

Пример

Нижеследующий цикл возводит в квадрат числа от 2 до 10:

```
int i =  2;
do
{
cout << i << " ^2  = " << i*i<<endl;
} while  (++i < 11);
```

Цикл while

Цикл while в C++ — второй условный цикл, в котором итерации вы полняются до тех пор, пока условие имеет значение true. Таким образом, цикл while может не выполнить ни одной итерации, если проверяемое условие изначально имеет значение false.

Общий синтаксис цикла while имеет вид:

```
while  (условие)
оператор; | { последовательность операторов }
```

Пример

```
// Вычисляет x в степени n
double pwr = 1;
while(n-- > 0)
  pwr *=  x;
cout << x << "^"<<pwr<<endl;
```

Пропуск итераций цикла

C++ предоставляет вам возможность сразу переходить к концу цикла и переходить к следующей итерации, используя оператор **continue**. Эта особенность позволяет вашему циклу пропускать итерации для отдельных значений, которые могут вызвать ошибки времени исполнения.

Общая форма использования оператора continue имеет вид:

```
<предложение начала цикла>
{
// последовательность операторов #1
 if (условие пропуска)
continue;
//  последовательность  операторов  #2
}   <предложение  конца цикла>
```

Выход из цикла

C++ поддерживает оператор **break** для выхода из цикла. Оператор **break** вызывает переход к выполнению кода, следующего за текущим циклом, подобно тому как это делается в операторе switch.

Общая форма использования оператора break в цикле имеет вид:

```
<предложение начала цикла>
{
// последовательность операторов #1
if (<условие выхода из цикла>)
break;
// последовательность операторов #2
} <предложение конца цикла>
// последовательность операторов #3
```

Вложенные циклы

Вложенные циклы позволяют вам представлять повторяющиеся задачи как часть других повторяющихся задач. C++ допускает вложение цикла любого типа на любую требуемую глубину. Вложенные циклы часто используются для обработки массивов.