Операторы управления циклом. Зацикливание

Существует два оператора, которые позволяют управлять выполнением цикла. Это операторы **break** и **continue**.

Давайте рассмотрим их работу на следующем примере: Модифицируем программу из прошлого урока.

> Программа: Игральный кубик.
>
> Программа заменяет обычный игральный кубик.
>
> Управление:
> 1 -- бросить кубик;
> 0 -- закончить игру.

В конце игры программа должна выводить количество бросков кубика, сделанных в игре. После **50** бросков программа автоматически завершается. Выводится сообщение **"Game over!"**.

Код такой программы будет выглядеть следующим образом:

Листинг 1.

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>

int main(void) {
  srand(time(NULL));

  printf("###########  Devil\'s bones   ###########\n");
  printf("#                                        #\n");
  printf("#   Commands:                            #\n");
  printf("#                                        #\n");
  printf("#   1 - new game                         #\n");
  printf("#   0 - exit                             #\n");
  printf("#                                        #\n");
  printf("##########################################\n\n");

  int ch_control;
  int value = 0, score = 0;

  do {
    printf("Input command: ");
    scanf("%d", &ch_control);

    switch(ch_control){
      case 0:
```

```
        score = score -1;
        break;
      case 1:
        value = 1 + rand()%6;
        printf("Result: %d\n", value);
        break;
      default:
        printf("Error! Try again...\n");
        continue;   // прерываем текущую итерацию цикла
    }

    score = score + 1;

    if(score == 50){
      printf("Game over!\n");
      break;       // завершаем цикл
    }
  }while(ch_control != 0);

  printf("\n\nSCORE: %d\n\nGood bye!\n", score);
  return 0;
}
```

Давайте поясню некоторые новые операторы, которые там используются.

Оператор break

Оператор **break** прекращает выполнение цикла. Помните, мы уже использовали данную команду в операторе **switch**. Здесь всё точно так же. Мы используем данный оператор, чтобы выйти из цикла, когда произойдёт **50** бросков.

Важный момент. Оператор **break** работает и во всех циклических конструкциях, и в операторе выбора.

В нашем примере он используется и там, и там. Возникает вопрос: а как он решает, откуда надо выйти: из **switch** или из цикла. Очень просто.

> Оператор **break** всегда завершает ближайший внешний оператор, внутри которого он находится.

Например, в нашей программе первый раз оператор **break** встречается в ветке **case 0**. Значит он находится внутри оператора **switch**, который находится внутри цикла **do-while**. Ближайший оператор, внутри которого он расположен, – это **switch**. Следовательно, завершается оператор **switch**. Аналогично и с другими ветками оператора **switch**.

Последний оператор **break** находится внутри **if**, который находится внутри цикла **do-while**. Т.к. на конструкцию **if** оператор **break** не действует, то ближайшим оператором, в котором он находится, является цикл **do-while**. Поэтому в этом случае **break** завершает цикл.

Оператор continue

Данный оператор применяется только внутри циклов. Он позволяет прервать текущую итерацию цикла. После того, как компьютер встречает оператор **continue**, он завершает исполнение тела цикла и переходит к проверке условия (в циклах **while** и **do-while**) или к изменению счётчика (выражение **3** в цикле **for**).

В нашей программе он используется для того, чтобы не учитывать плохие ходы. Например, пользователь ввёл число **2**. Программа выдаёт ему сообщение об ошибке и сразу же переходит к проверке условия. При этом все команды ниже пропускаются, а значит не увеличивается счётчик ходов, и не проверяется превышение предела в **50** ходов на игру. Если здесь убрать оператор, то после вывода сообщения об ошибке тело цикла продолжит выполняться дальше и увеличится счётчик ходов.

Зацикливание

Иногда бывает так, что условие, при котором работа цикла должна прекратиться, никогда не выполняется. В таких случаях говорят, что программа «зациклилась».

Зачастую это происходит при использовании циклов **do-while** и **while**.

Пример: программа сложения пяти введённых чисел.

Листинг 2.

```
#include <stdio.h>

int main(void) {
  double sum = 0, temp = 0;
  int k = 0;

  do {
    scanf("%lf", &temp);
    sum = sum + temp;
  } while(k < 5);

  printf("%.3f", sum);
  return 0;
}
```

На первый взгляд всё верно, но попробуйте скомпилировать и запустить эту программу. Вы заметите, что она не спешит останавливаться после того, как мы ввели первые пять чисел.

Когда мы используем эти циклы, необходимо внимательно отслеживать, что переменные, входящие в условия цикла, в теле цикла хоть как-то изменяются.

В нашем примере допущена именно такая ошибка. Условие записано верно, переменная **k** (счётчик считанных чисел) объявлена и инициализирована, но внутри тела

цикла мы забыли её увеличить. Поэтому нашему условию окончания цикла `k < 5` не суждено стать истинным.

Не всегда подобные ошибки так очевидны, как в нашем случае. Поэтому будьте внимательны, когда используете циклы с условиями.

Давайте приведем нашу программу в рабочее состояние:

Листинг 3.

```
#include <stdio.h>

int main(void) {
  double sum = 0, temp = 0;
  int k = 1;

  do {
    scanf("%lf", &temp);
    sum = sum + temp;
    k = k + 1;
  } while(k <= 5);

  printf("%.3f", sum);
  return 0;
}
```

Дополнительные материалы

1. Иногда бесконечные циклы могут использоваться преднамеренно. Например, при создании ботов или в олимпиадном программировании для экономии времени. Подробнее об этом в Википедии

Интернет версия: http://youngcoder.ru/lessons/7/upravlenie_ciklom.php