Циклы с условиями

Иногда заранее нельзя предсказать, какое количество раз должен выполниться цикл. Но при этом известно некоторое условие, когда цикл должен остановиться. Например:

Программа: Игральный кубик.

Программа заменяет обычный игральный кубик.

Управление:
1 -- бросить кубик;
0 -- закончить игру.

Давайте напишем заготовку для нашей игры.

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 control;
  int value = 0;
  printf("Enter command: ");
  scanf("%d", &control);

  if(control == 1){
    value = 1 + rand()%6;
    printf("Result: %d\n", value);
  }

  return 0;
}
```

Тут-то мы и сталкиваемся с проблемой. Понятно, что заранее узнать, когда игра закончится невозможно. А значит использовать цикл **for** напрямую не получится. Одним из выходов из подобной ситуации является использование других циклических конструкций. Например, цикла **while**.

Цикл с предусловием while

Листинг 2.

```
while (условие)
  оператор;
```

Работает эта конструкция следующим образом:

1. Программа встречает ключевое слово **while**, значит дальше идёт циклическая конструкция;
2. Проверяется условие. Вычисляется логическое выражение, записанное в скобках;
3. Если значение условия **ИСТИНА**, то выполняется тело цикла. Переходим к пункту 2;
4. Если значение условия **ЛОЖЬ**, то цикл завершается. Управление передаётся на оператор, следующий за телом цикла.

Под оператором понимается один оператор. Если нужно выполнить в цикле несколько команд, то необходимо использовать составной оператор **{}**.

Давайте перепишем нашу программу с использованием данного цикла.

Листинг 3.

```
#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 control;
  int value = 0;

  printf("Enter command: ");
  scanf("%d", &control);
  while(control != 0){
    switch(control){
      case 1:
        value = 1 + rand()%6;
```

```
        printf("Result: %d\n", value);
        break;
      default:
        printf("Error! Try again...\n");
        break;
    }
    printf("Enter command: ");
    scanf("%d", &control);
  }

  printf("Good bye!\n");
  return 0;
}
```

Опишем словами алгоритм работы данной программы:

1. Выводим меню пользователя и предложение ввести команду;
2. Считываем код команды в переменную **control**;
3. Запускаем цикл **while**. Проверяем условие;
4. Если пользователь ввёл **0**, то условие выполнения цикла принимает значение **ЛОЖЬ**. Тело цикла не выполняется. Управление передаётся на следующий за циклом оператор. Выводится строка **Good bye!**. Программа завершается;
    1. Оператор выбора:
        1. Если пользователь ввёл **1**, то генерируем случайное число от **1** до **6** и выводим его на экран. Выходим из оператора выбора;
        2. Если пользователь ввёл что-то иное, выводим сообщение об ошибке. Выходим из оператора выбора.
    2. Выводим пользователю предложение ввести новую команду;
    3. Считываем код команды в переменную **control**;
    4. Возвращаемся к проверке условия. Пункт 3.

Цикл **while** называют циклом с предусловием, т.к. прежде, чем выполнить тело цикла, проверяется условие. Это значит, например, что возможна такая ситуация, что тело цикла не выполнится вообще ни один раз. Другое название цикла **while** – цикла **ПОКА**. Дословный перевод с английского. Это название отображает саму суть цикла.

Мнемоническое правило:
ПОКА условие ИСТИНА, выполняй тело цикла.

Цикл с постусловием do-while

И последняя, третья циклическая конструкция – цикл **do-while**.

Данный цикл отличается от цикла **while** тем, что условие проверяется не перед выполнением тела цикла, а после выполнения тела цикла. Это значит, что тело цикла **do - while** выполнится хотя бы один раз обязательно.

Синтаксис данной циклической конструкции таков:

Листинг 4.

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

Работает эта конструкция следующим образом:

1. Программа встречает ключевое слово **do**. Значит перед ней цикл **do-while**;
2. Выполняется тело цикла;
3. Проверяется условие;
4. Если условие **ИСТИНА**,то снова выполняется тело цикла;
5. Если условие **ЛОЖЬ**, то работа циклической конструкции прекращается. Программы выполняет оператор, следующий за циклом **do-while**.

Давайте перепишем нашу программу с использованием данного типа циклической конструкции.

Листинг 5.

```
#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;

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

    switch(ch_control){
      case 0: break;
      case 1:
        value = 1 + rand()%6;
        printf("Result: %d\n", value);
        break;
      default:
        printf("Error! Try again...\n");
        break;
    }
  }while(ch_control != 0);

  printf("Good bye!\n");
  return 0;
}
```

В общем-то очень похоже на предыдущий код. Правда, пришлось немножко поменять оператор выбора: добавить туда ветку **`case 0:`**. Иначе из-за того, что проверка производится после выполнения тела цикла, программа работала некорректно. При вводе нуля появлялось сообщение об ошибке. В прошлой программе (с циклом **`while`**) подобной ситуации быть не могло, т.к. равенство нулю проверялось в условии цикла. При вводе нуля условие стало бы ложью, а значит цикл завершился бы и тело цикла не выполнилось.

Практика

1. Решите предложенные задачи с автоматической проверкой решения.

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