Семинар 5. Циклы, часть 1

Вовочка плохо себя вел на уроке и в наказание должен был написать на доске 100 раз «Я не буду больше пускать самолетики в классе». Но он не расстроился, вышел к доске и справился с поставленной задачей за минуту:

```
for(i=1; i<=100; i++)
{
      Console.WriteLine("Я не буду больше пускать самолетики");
}
```

На прошлом занятии мы научились разветвлять алгоритмы, теперь пришло время рассмотреть остальные способы управления последовательностью выполнения команд, наиболее важными из которых являются циклы.

Циклы необходимы в тех случаях, когда необходимо выполнить похожие действия несколько раз подряд. Существуют различные виды циклов, и программисту приходится выбирать наиболее подходящий для каждого конкретного случая. Начнем с самого простого.

5.1 Цикл for

Цикл for используется, когда необходимо выполнить какое-то конкретное действие определенное количество раз. Например, напечатать на экране 100 раз одно и ту же фразу.

```
Цикл из одного действия:
for (инициализация; условие; изменение) действие;

Цикл, тело которого состоит из нескольких операторов:
for (инициализация; условие; изменение) действие
{
      // несколько действий выполняемых подряд
      Оператор1;
      Оператор2;
      Оператор3;
}
```

Действие будет выполняться до тех пор, пока условие будет правдой и прекратится сразу же, как только условие станет ложью. В любом месте цикла можно его разорвать с помощью команды «break» или перейти к следующей итерации в цикле с помощью «continue», но такой подход не приветствуется, т.к. в этом случае становится очень сложно понять структуру алгоритма.

Действия, выполняемые циклом, называются его телом. Каждый проход цикла, по-умному, называется итерация. Если же нужно, циклически повторить несколько операторов, их необходимо взять в фигурные скобки.

Понять принцип действия любой конструкции проще всего на практике. К примеру, давай выведем на экран все целые числа от 1 до 100. Можно конечно долго и утомительно писать что-то вроде

```
Console.Write («1»);
Console.Write («2»);
Console.Write («3»);
...
```

Представь себе, как это долго и утомительно! Но к счастью у нас есть цикл for.

```
for(int i=1; i <= 100; i++) Write(i);
```

Всю программу мы уместили в одну строчку, вместо ста! Неплохо, не правда ли? Теперь давай разберемся, что же мы такого натворили!

Со слова «for», понятное дело, начинает цикл. Далее, в скобках через точку с запятой, перечислено три параметры цикла:

- Первый параметр – инициализация счетчика. В нем мы должны присвоить счетчику его начальное значение. Если переменная счетчика ранее не была описана, необходимо её описать прямо в параметре, как, и сделано в нашем примере. Мы начали цикл с единицы, хотя имеем право начать его совершенно с любого числа.
- Второй параметр – условие. Цикл будет выполняться до тех пор, пока условие будет верным.
- Третий параметр – изменение счетчика. В нем нужно описать, как будет изменяться переменная счетчика каждую итерацию. Обычно, счетчик увеличивают или уменьшают на единицу, но можно сделать и так, чтобы счетчик изменялся с некоторым шагом, например i+=10.

Соответственно, цикл завершится, когда переменная счетчика выйдет за пределы, неудовлетворяющие заданному условию. В нашем случае – когда он станет равным 101.

5.2 Цикл while

Общий вид этой конструкции следующий:

```
Цикл из одного действия:
while (условие) действие;

Цикл, тело которого состоит из нескольких операторов:
while (условие)
{
      // несколько действий выполняемых подряд
      Оператор1;
      Оператор2;
      Оператор3;
}
```

Оператор while работает по следующему принципу: сначала проверяется выражение и, если оно истинно, выполняется тело цикла. Если же условие ложно, цикл не выполняется вообще. Сразу после выполнения тела цикла программа возвращается к условию и все повторяется. Приведем пример из реальной жизни. Квест, как попасть домой

```
while (дверь_закрыта) звонить_в_дверной_звонок;
зайти_в_дверь;
```

Действуя в рамках цикла, мы будем бесконечно «звонить в дверной звонок», пока она не откроется. Открытие двери превращает условие «дверь_закрыта» в ложь, цикл сразу же завершает свою работу и управление переходит к оператору «зайти_в_дверь». Вот такая вот логика!

5.3 Примеры использования циклов (программа «Проверка числа»)

Следующая наша задача будет в том, чтобы определить, является ли число, введенное пользователем простым.

Вспомним из математики, что простые числа – это те, которые делятся без остатка только на единицу и на само себя, например, 2, 3, 5, 7, 11, 13, 17, 19. Число 4 уже не является простым, так как делится не только на 1 и 4, но еще и на 2.

Допустим, нам дано число N. Тогда для решения поставленной задачи нам нужно будет в цикле проверить делимость заданного числа на все целые числа от 2 до N-1. Для этого воспользуемся оператором «%», которые позволяет получить остаток от деления.

```
Int32 number=0; String msgtext = "";
Console.WriteLine("Введи число: ");
number = Convert.ToInt32(Console.ReadLine());
for (Int32 i=2; i<=number-1; i++)
{
      if (number % i == 0)
            msgtext = "Число "+number+" составное.";
}
msgtext = (msgtext!="") ? msgtext : "Число "+number+" простое!";
Console.WriteLine(msgtext);
```

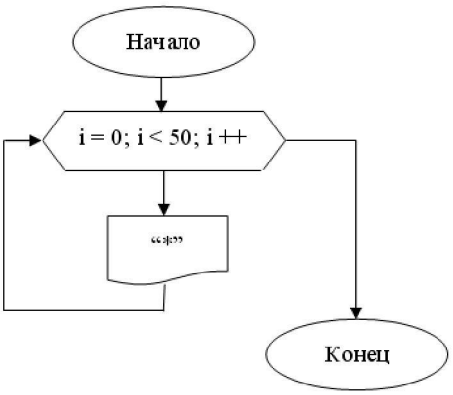
Задания

#1 Написать программу, выводящую на экран таблицу квадратов четных чисел от 0 до 10 в убывающем порядке: 100, 81, 49…

#2 Составь блок-схему алгоритма, и напиши программу которая выводит на экране узор, состоящий из звездочек. Должно быть 20 строчек со звездочками так, чтобы в каждой последующей строке было на одну звездочку больше. Для вывода на экран строки, содержащей N звездочек, используй команды

String repeatedString = new String('*', N);

Console.Write (repeatedString);



#3 Напиши программу, которая выводит сумму N чисел, введенных пользователем. Число N должно так же определяться пользователем.

#4 Напиши игрушку для тренировки устного счета. Пользователю 10 раз должны предлагаться случайные два числа в диапазоне от 0 до 20, которые он должен будет перемножить. Если ответ правильный, программа выводит сообщение с поздравлением и следующий пример, если пользователь ошибется, тогда сообщение

об ошибки и так же следующий пример. В конце игры пользователь получает сумму набранных баллов.

#5 Напиши программу, которая «задумывает» число в диапазоне от 1 до 10 и предлагает пользователю угадать его за 5 попыток. После каждой попытки программа говорит, что названное число больше загаданного или меньше.