МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ КЫРГЫЗСКОЙ РЕСПУБЛИКИ

ОШСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ МАТЕМАТИКИ И ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

КАФЕДРА ПРОГРАММИРОВАНИЯ

Разработка лекционного занятия по дисциплине «Разработка приложений на Visual Studio 2010»

Тема лекции: **Циклы while. Циклы for. Прерывание циклов.**

Группа: **ИСТ(м)-1-19**

Лектор: **доц.каф. «Программирования», Аркабаев Н.К.**

Ош – 2019

<u>Тема занятия:</u> **Циклы while. Циклы for. Прерывание циклов**

Тип урока: объяснение нового материала с выполнением практической работы.

Цель: познакомить с понятием цикла, видами циклических алгоритмов, сформировать умения пользоваться операторами цикла, сформировать умение решать задачи с использованием циклов. Обеспечить прочное, сознательное овладение магистрантов основами знаний об организации циклов в среде программирования Visual Studio C#. Работа с прерыванями циклов.

Задачи:

- **Обучающие:** познакомить учащихся с применением циклов в языке ObjectPascal, закрепление навыков грамотности написания операторов языка OPascal; овладение синтаксисом циклических конструкций, умением строить блок-схем алгоритмов задач с циклами; показать сходство и различие операторов цикла в языке программирования OPascal. Овладение современным стилем программирования. Выполнение заданий творческого характера, требующих системного, исследовательского подхода к решению проблемного вопроса.
- **Развивающие**: Развитие памяти, внимания, познавательного интереса у учащихся, умения обобщать, анализировать, сравнивать, использовать накопленные ранее знания для решения практических задач. Развитие алгоритмического и логического мышления, умение правильно сформулировать математическую модель и алгоритм решения поставленной задачи. Активизация взаимодействия между учащимися, развитие навыков командной работы.
- **Воспитательные:** Воспитание познавательной потребности, интереса к предмету. Воспитание умения слушать друг друга, культуры речи, взаимовыручки, ответственности в ходе командной игры. Воспитание интереса к учению и формирование познавательной активности. Воспитывать умение отстаивать свою точку зрения, аргументировано вести диалог с одногруппниками.
- **Методические:** Показать методику проведения урока с использованием игровых методов командной работы. Показать вариант применения флипчарта на базе SmartPodiuma. Продемонстрировать метод использования опорных конспектов и формативного метода оценивания.

Методы и приемы: Элементы критического мышления, наглядный, диалогический, самостоятельная работа.

Средства: устная речь, письмо, задания для индивидуальной и групповой работы.

ТСО: SmartPodium, среда программирования Visual Studio C#, программное обеспечение «Visual Studio 2010», презентация «PowerPoint», лллекции по теме «Презентация лекции. Операторы цикла».

Дидактическое обеспечение: флипчарт, рабочие листы, интерактивная доска, тестовые материалы в системе kahoot!.

Межпредметные связи: Архитектура информационных систем, программирование Web-ориентированных приложений, математика, английский язык, методы компьютерного моделирования.

Методическое обеспечение урока: 1) Г. Шилдт. Полный справочник по C#. : Пер. с англ. - М. : Издательский дом "Вильяме", 2004. - 752 с.

2) 2. Джесс Либерти. Программирование на C#.: Пер. с англ. – 2-е изд, С.Пб.: Издательство «Символ-плюс», 2002. – 684 с.

Этапы занятия:

1. Сообщение темы, цели задач урока и мотивация учебной деятельности (2 мин.)
2. Подготовка к изучению нового материала через повторение и актуализацию опорных знаний (8 мин.)
3. Изучение нового материала: (30 мин.)
   - циклы while;
   - циклы for;
   - прерывание циклов.
4. Первичное осмысление и закрепление связей и отношений в объектах изучения в системе kahoot! (5 мин.)
5. Итог урока (2 мин.)
6. Постановка задания на дом (1 мин.)
7. Рефлексия (1 мин)

Теоретический материал урока.

Циклы. Циклическое выполнение означает многократное выполнение операторов. Это наиболее мощная возможность во всем программировании, поскольку она позволяет выполнять необходимые операции столько раз, сколько требуется, без повторного написания одного и того же кода.

В качестве простого примера рассмотрим следующий код, вычисляющий сумму денег на банковском счету через 10 лет при условии ежегодной выплаты процентов без снятий и поступлений средств:

```
double balance =1000;
double interestRate = 1.05; // годовая ставка 5%
balance *= interestRate; balance *= interestRate;
balance *= interestRate; balance *= interestRate;
```

```
balance *= interestRate; balance *= interestRate;
balance *= interestRate; balance *= interestRate;
```

Десятикратная запись одной и той же строки сама по себе неэффективна, а если понадобится заменить срок 10 лет каким-нибудь другим значением? Тогда придется вручную копировать эту строку кода требуемое число раз, что весьма утомительно. К счастью, это совсем необязательно. Вместо этого можно создать цикл, выполняющий необходимую инструкцию нужное количество раз.

Имеется еще один важный вид циклов, выполняющийся до тех пор, пока не будет удовлетворено определенное условие. Такие циклы выглядят немного проще, чем вышеописанные (хотя и не менее полезны), поэтому давайте с них и начнем.

Циклы do

Циклы do действуют следующим образом. Сначала выполняется код тела цикла, затем производится проверка логического условия, и если она возвращает true, то тело цикла выполняется снова — и так до тех пор, пока проверка не возвратит false, после чего цикл завершается. Ниже приведен синтаксис цикла do; подразумевается, что выражение <проверка> должно возвращать одно из булевских значений:

```
do
{
<код тела цикла>
} while (<проверка>) ;
```

Символ точки с запятой после оператора `while` обязателен

Например, для вывода чисел от 1 до 10 в столбик можно использовать такой цикл do:

```
int i = 1;
do
{
Console .WriteLine (" {0} ’’, i++); } while (i <= 10);
```

Здесь постфиксная версия операции ++ увеличивает значение i на 1 после его вывода на экран, поэтому проверка должна иметь вид i <= 10, чтобы число 10 также было выве¬дено на консоль.

В следующем практическом занятии похожий код выводит результат вычисления ба¬ланса на счету по прошествии количества лет, необходимого для накопления на этом счету указанной суммы денег, исходя из заданной начальной суммы и фиксированной процент¬ной ставки.

Применение циклов do.

Создайте новое консольное приложение с именем ChO4ExO4 и сохраните его в ката¬логе C:\BegVCSharp\ChapterO4.

Добавьте в файл Program.cs следующий код:

static void Main (string [ ] args)

{

double balance, interestRate, targetBalance;

Console.WriteLine("What is your current balance?");

// Каков текущий баланс? balance = Convert.ToDouble(Console.ReadLine()); Console.WriteLine("What is your current annual interest rate (in %)?");

// Какова ежегодная ставка (в процентах)? interestRate = 1 + Convert.ToDouble(Console.ReadLine O) / 100.0; Console.WriteLine("What balance would you like to have?");

// Какой баланс необходимо получить? targetBalance = Convert.ToDouble(Console.ReadLine()); int totalYears = 0;

do

{

balance *= interestRate;

++totalYears;

)

while (balance < targetBalance);

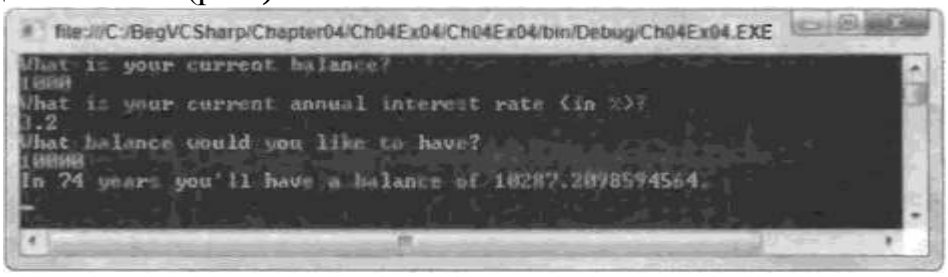
Console.WriteLine("In {0} year(l} you'll have a balance of {2}.", totalYears, totalYears == 1 ? "" :’"s", balance);

// Вывод баланса через вычисленное количество лет Console.ReadKey();

}

Фрагмент кода Ch04Ex04\Program. cs

Запустите это приложение и в ответ на приглашение введите какие-нибудь значения (рис.).



Описание работы.

В приведенном коде операция ежегодного подсчета баланса с фиксированной процент¬ной ставкой просто повторяется такое количество раз, которое необходимо, чтобы баланс стал удовлетворять конечному условию. Подсчет количества необходимых для этого лет ведется путем увеличения на 1 значения переменной-счетчика в каждой итерации цикла: int totalYears = 0;

do

{

balance *= interestRate;

++totalYears;

}

while (balance < targetBalance);

После этого значение данной переменной-счетчика можно использовать как часть выводимого на экран результата:

Console.WriteLine ("In {0} year{1} you’ll have a balance of {2}.”,
totalYears, totalYears == 1 ? ”” : "s", balance) ;

Циклы do всегда выполняются как минимум один раз. Иногда, как, например, в данной ситуации, такой вариант нежелателен. Разумеется, можно добавить оператор if:

```
int totalYears = 0;
if (balance < targetBalance)
{
do
{
balance *= interestRate; ++totalYears;
}
while (balance < targetBalance);
}
Console.WriteLine("In {0} year(l) you'll have a balance of {2}.", totalYears,
totalYears == 1 ? "" : "s", balance);
```

Очевидно, что это все-таки усложняет код. Поэтому гораздо лучшим решением является цикл while.

Циклы while

Циклы while очень похожи на циклы do, но имеют одно важное отличие: логическая проверка в них выполняется в начале, а не в конце цикла. Если проверка сразу возвращает false, код тела цикла вообще не выполняется, а поток выполнения переходит на код, следующий за циклом.

>

Синтаксис цикла while выглядит следующим образом:

while (<проверка>)

Выполнение практического задания.

А теперь давайте закрепим изученный материал. Переместитесь, пожалуйста, за компьютеры.

Подведение итогов урока:

Сегодня мы с вами выяснили, как работать с циклами в среде Visual Studio C#. Каковы ее назначение и состав. Познакомились с различными приемами для работы с циклами, с прерываниями циклов. На следующем уроке мы подробно рассмотрим работы с вложенными циклами и рекурсией.

Спасибо за урок. До свидания.