Лабораторная работа №1

Основы программирования на Visual C# .NET

Язык программирования C# является прямым наследником языка C++. Он унаследовал многие синтаксические конструкции языка C и объектно-ориентированную модель C++. В отличие от C++ C# является чисто объектно-ориентированным языком. В объектно-ориентированном программировании ход выполнения программы определяется объектами. Объекты это экземпляры класса. Класс это абстрактный тип данных, определяемый пользователем (программистом). Класс включает в себя данные и функции для обработки этих данных. В C# запрещены глобальные функции. Все функции должны быть обязательно определены внутри класса. Не является исключением и главная функция языка C# Main( ) (в отличии от языка C пишется с прописной буквы).

Объявление класса синтаксически имеет следующий вид:

```
class имя_класса
        {
                // члены класса
        }
```

Члены класса это данные и функции для работы с этими данными. Не затрагивая пока определения и свойств членов класса, рассмотрим общую структуру приложения для консоли. Для этого создадим пустой проект в интегрированной рабочей среде Microsoft Visual Studio.NET и проанализируем, что подготовит нам мастер приложения.

Запустим на выполнение среду Microsoft Visual Studio.NET через кнопку «Пуск» панели задач. В открывшемся окне выберем пункт File|New|Project.

В диалоговом окне New Project необходимо выбрать тип проекта Visual C# Projects и шаблон приложения - Console Application.

В поле наименование (Name) ввести новое имя проекта, либо согласиться с именем, которое предлагает мастер.

В поле местоположение (Location) выбрать или ввести полный путь к рабочей папке проекта.

При этом появится шаблон приложения, подготовленный для нас мастером:

```
using System;
namespace ConsoleApplication10
{
        /// <summary>
        /// Summary description for Class1.
        /// </summary>
        class Class1
        {
                /// <summary>
                /// The main entry point for the application.
                /// </summary>
                [STAThread]
                static void Main(string[] args)
```

```
        {
            //
            // TODO: Add code to start application here
            //
        }
    }
}
```

Первая строчка проекта *using System;*, включает в себя директиву *using*, которая сообщает компилятору, где он должен искать классы (типы), не определенные в данном пространстве имен. Мастер, по умолчанию, указывает стандартное пространство имен *System*, где определена большая часть типов среды .NET.

Следующей строчкой *namespace ConsoleApplication10* мастер предложения определяет пространство имен для нашего приложения. По умолчанию в качестве имени выбирается имя проекта. Область действия пространства имен определяется блоком кода, заключенного между открывающей и закрывающей фигурными скобками. Пространство имен обеспечивает способ хранения одного набора имен отдельно от другого. Имена, объявленные в одном пространстве имен не конфликтуют, при совпадении, с именами, объявленными в другом пространстве имен.

В шаблоне приложения имеется множество строк, которые являются комментариями.

В C# определены три вида комментариев:

- многострочный (/*…*/)
- однострочный (//…)
- XML (///) – комментарий для поддержки возможности создания самодокументированного кода.

Строчка [STAThread] является атрибутом. Атрибуты задаются в квадратных скобках. С помощью атрибута в программу добавляется дополнительная описательная информация, связанная с элементом кода, непосредственно перед которым задается атрибут. В нашем случае

указывается однопоточная модель выполнения функции Main. Заголовок функции:

```
static void Main(string[] args)
```

Функция Main определена как статическая (static) с типом возвращаемого значения void. Функция Main( ) C# как и функция main( ) языка C может принимать аргументы. Аргумент - это строковый массив, содержащий элементы командной строки. Тело функции пустое и в нем содержится, в виде комментария, предложение добавить туда код для запуска приложения:

```
// TODO: Add code to start application here
```

Воспользуемся этим предложением и добавим в тело функции одну строчку:

```
static void Main(string[] args)
{
    //
    // TODO: Add code to start application here
```

```
            Console.WriteLine("Привет!");
            //
        }
```

затем, скомпилируем и запустим приложение на выполнение. Для этого необходимо выбрать пункт меню **Debug | Start Without Debugging** или нажать комбинацию клавиш **Ctrl+F5**. В результате выполнения программы появится окно:

Строка:

Console.WriteLine("Привет!");,

выводит сообщение на консоль.

Функции консольного ввода-вывода являются методами класса Console библиотеки классов среды .NET.

Для ввода строки с клавиатуры используется метод Console.ReadLine(), а для ввода одного символа метод Console.Read().

Для консольного вывода также имеются две метода

- метод Console.Write(), который выводит параметр, указанный в качестве аргумента этой функции, и
- метод Console.WriteLine(),который работает так же, как и Console.Write(), но добавляет символ новой строки в конец выходного текста.

Для анализа работы этих методов модифицируйте функцию Main( ) так, как показано ниже :

```
        static void Main(string[] args)
        {
            //
            // TODO: Add code to start application here
            Console.WriteLine("Введите ваше имя");
            string str=Console.ReadLine();
            Console.WriteLine("Привет "+str+"!!!");
            Console.WriteLine("Введите  один символ с клавитуры");
            int kod=Console.Read();
            char sim=(char)kod;
            Console.WriteLine("Код символа  "+sim+" = "+kod);

            //
        }
```

Скомпилируйте и выполните приложение.

Проанализируйте код функции Main( ) и результат ее работы.

Здесь хорошо видно, что строку вывода метода **Console.WriteLine** можно формировать из переменных разного типа, просто объединяя их в одну строку с помощью знака +.

Можно сформировать точно такую же выходную строку, используя другую модификацию метода **Console.WriteLine** , которая принимает несколько параметров (список параметров), наподобие функции **printf()** в **C**. Первым параметром списка является строка, содержащая маркеры в фигурных скобках. Маркер это номер параметра в списке. При выводе текста вместо маркеров будут подставлены соответствующие параметры из остального списка.

Для демонстрации работы этого метода вставьте в конец кода программы, строчку, как показано на листинге ниже.

```
static void Main(string[] args)
        {
            //
            // TODO: Add code to start application here
            Console.WriteLine("Введите ваше имя");
            string str=Console.ReadLine();
            Console.WriteLine("Привет "+str+"!!!");
            Console.WriteLine("Введите  один символ с клавитуры");
            int kod=Console.Read();
            char sim=(char)kod;
            Console.WriteLine("Код символа  "+sim+" = "+kod);
            Console.WriteLine("Код символа  {0} = {1}",sim,kod);

            //
        }
```

Скомпилируйте и выполните это приложение. Проанализируйте, полученный результат.

После маркера через запятую можно указать, сколько позиций отводится для вывода значений. Например, запись {1,3} означает, что для печати первого элемента списка отводится поле шириной в три символа. Причем, если значение ширины положительно, то производится выравнивание по правому краю поля, если отрицательно то по левому.

Добавив 4 новые строчки в конец кода функции Main(), убедимся в этом:

```
static void Main(string[] args)
  {
  //
  // TODO: Add code to start application here
        Console.WriteLine("Введите ваше имя");
        string str=Console.ReadLine();
        Console.WriteLine("Привет "+str+"!!!");
Console.WriteLine("Введите  один символ с клавитуры");
        int kod=Console.Read();
        char sim=(char)kod;
```

```
        Console.WriteLine("Код символа  "+sim+" = "+kod);
        Console.WriteLine("Код символа  {0} = {1}",sim,kod);
        int s1=255;
        int s2=32;
        Console.WriteLine(" \n{0,5}\n+{1,4}\n-----\n{2,5}",s1,s2,s1+s2);
        Console.WriteLine(" \n{1,5}\n+{0,4}\n-----\n{2,5}",s1,s2,s1+s2);
  //
  }
```

Кроме того, после поля ширины через двоеточие можно указать форматную строку, состоящую из одного символа и необязательного значения точности.

Существует 8 различных форматов вывода:

C – формат национальной валюты,

D – десятичный формат,

E – научный (экспоненциальный) формат,

F – формат с фиксированной точкой,

G – общий формат,

N – числовой формат,

P – процентный формат,

X – шестнадцатеричный формат

Например, запись {2,9:C2} – означает, что для вывода второго элемента из списка, отводится поле шириной в 9 символов. Элемент выводится в формате денежной единицы с количеством знаков после запятой равной двум. При выводе результата происходит округление до заданной точности.

Для иллюстрации вышесказанного модифицируем функцию Main(), как показано ниже:

```
static void Main(string[] args)
  {
  //
  // TODO: Add code to start application here
        Console.WriteLine("Введите ваше имя");
        string str=Console.ReadLine();
        Console.WriteLine("Привет "+str+"!!!");
Console.WriteLine("Введите  один символ с клавитуры");
        int kod=Console.Read();
        char sim=(char)kod;
        Console.WriteLine("Код символа  "+sim+" = "+kod);
        Console.WriteLine("Код символа  {0} = {1}",sim,kod);
        int s1=255;
        int s2=32;
        Console.WriteLine(" \n{0,5}\n+{1,4}\n-----\n{2,5}",s1,s2,s1+s2);
        Console.WriteLine(" \n{1,5}\n+{0,4}\n-----\n{2,5}",s1,s2,s1+s2);
        double sum1=500.3467;
```

```
            double sum2=43.5;
            Console.WriteLine(" \n{0,10:C2}\n+{1,9:C2}\n-----\n{2,10:C2}",
                  sum1,sum2,sum1+sum2);

        //
        }
```

Скомпилируйте и запустите код на выполнение.

Проанализируйте изменения, происходящие с выходным текстом, при применении различных форматов.

Задания для лабораторной работы:

Вариант 1

1. Построить описание класса, содержащего информацию о почтовом адресе организации. Предусмотреть возможность раздельного изменения составных частей адреса, создания и уничтожения объектов этого класса.

2. Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа unsigned char — для копеек. Дробная часть (копейки) при выводе на экран должна быть отделена от целой части запятой. Реализовать сложение, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и операции сравнения.

Вариант 2

1. Составить описание класса для представления комплексных чисел с возможностью задания вещественной и мнимой частей как числами типов double, так и целыми числами. Обеспечить выполнение операций сложения, вычитания и умножения комплексных чисел.

2. Создать класс Triangle для представления треугольника. Поля данных должны включать углы и стороны. Требуется реализовать операции: получения и изменения полей данных, вычисления площади, вычисления периметра, вычисления высот, а также определения вида треугольника (равносторонний, равнобедренный или прямоугольный).

Вариант 3

1. Составить описание класса для работы с цепными списками строк (строки произвольной длины) с операциями включения в список, удаления из списка элемента с заданным значением данного, удаления всего списка или конца списка, начиная с заданного элемента.

2. Создать класс Angle для работы с углами на плоскости, задаваемыми величиной в градусах и минутах. Обязательно должны быть реализованы: перевод в радианы, приведение к диапазону 0-360, увеличение и уменьшение угла на заданную величину, получение синуса, сравнение углов.

Вариант 4

1. Составить описание класса для объектов – векторов, задаваемых координатами концов в трехмерном пространстве. Обеспечить операции сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между векторами

1. Создать класс Point для работы с точками на плоскости. Координаты точки — декартовы. Обязательно должны быть реализованы: перемещение точки по оси ***X,*** перемещение по оси ***Y,*** определение расстояния до начала координат, расстояния между двумя точками, преобразование в полярные координаты, сравнение на совпадение и несовпадение.

Рациональная (несократимая) дробь представляется парой целых чисел (*a, b*), где *a* — числитель, *b* — знаменатель. Создать класс Rational для работы с рациональными дробями. Обязательно должны быть реализованы операции:

- ***сложения add,*** *(a, b)* **+** ***(c,*** *d) = (ad + bc, bd);*
- вычитания sub, *(a,* ***b)*** - *(c,* ***d) = (ad - bc, bd);***
- умножения mul, (a, ***b)*** * (c, ***d) = (ac, bd);***

- ***деления div,*** *(a, b) /* **(c,** *d) = (ad, bc);*
- сравнения equal, greate, less.

Должна быть реализована приватная функция сокращения дроби Reduce, которая обязательно вызывается при выполнении арифметических операций

Вариант 5

1. Составить описание класса прямоугольников со сторонами, параллельными осям координат. Предусмотреть возможность перемещения прямоугольников на плоскости, изменения размеров, построения наименьшего прямоугольника, содержащего два заданных прямоугольника, и прямоугольника, являющегося общей частью (пересечением) двух прямоугольников.

2. Создать класс Date для работы с датами в формате «год.месяц.день». Дата представляется структурой с тремя полями типа unsigned int: для года, месяца и дня. Класс должен включать не менее трех функций инициализации: числами, строкой вида «год.месяц.день» (например, «2004.08.31») и датой. Обязательными операциями являются: вычисление даты через заданное количество дней, вычитание заданного количества дней из даты, определение високосности года, присвоение и получение отдельных частей (год, месяц, день), сравнение дат (равно, до, после), вычисление количества дней между датами.

Вариант 6

1.Составить описание класса для определения одномерных массивов целых чисел (векторов). Предусмотреть возможность обращения к отдельному элементу массива с контролем выхода за пределы индексов, возможность задания произвольных границ индексов при создании объекта и выполнения операций поэлементного сложения и вычитания массивов с одинаковыми границами индексов, умножения и деления всех элементов массива на скаляр, печати (вывода на экран) элементов массива по индексам и всего массива.

2. Создать класс Time для работы со временем в формате «час:минута:секунда». Класс должен включать в себя не менее четырех функций инициализации: числами, строкой (например, «23:59:59»), секундами и временем. Обязательными операциями являются: вычисление разницы между двумя моментами времени в секундах, сложение времени и заданного количества секунд, вычитание из времени заданного количества секунд, сравнение моментов времени, перевод в секунды, перевод в минуты (с округлением до целой минуты).

Вариант 7

1. Составить описание класса для определения одномерных массивов строк фиксированной длины. Предусмотреть возможность обращения к отдельным строкам массива по индексам, контроль выхода за пределы индексов, выполнения операций поэлементного сцепления двух массивов с образованием нового массива, слияния двух массивов с исключением повторяющихся элементов, печать (вывод на экран) элементов массива и всего массива.

2. Реализовать класс Account, представляющий собой банковский счет. В классе должны быть четыре поля: фамилия владельца, номер счета, процент начисления и сумма в рублях. Открытие нового счета выполняется операцией инициализации. Необходимо выполнять следующие операции: сменить владельца счета, снять некоторую сумму денег со счета, положить деньги на счет, начислить проценты, перевести сумму в доллары, перевести сумму в евро, получить сумму прописью (преобразовать в числительное).

Вариант 8

1. Составить описание класса многочленов от одной переменной, задаваемых степенью многочлена и массивом коэффициентов. Предусмотреть методы для вычисления значения многочлена для заданного аргумента, операции сложения, вычитания и умножения многочленов с получением нового объекта-многочлена, печать (вывод на экран) описания многочлена.

2. Создать класс Goods (товар). В классе должны быть представлены поля: наименование товара, дата оформления, цена товара, количество единиц товара, номер накладной, по которой товар поступил на склад. Реализовать методы изменения цены товара, изменения количества товара (увеличения и уменьшения), вычисления стоимости товара. Метод toStringO должен выдавать в виде строки стоимость товара.

Вариант 9

1. Составить описание класса одномерных массивов строк, каждая строка которых задается длиной и указателем на выделенную для нее память. Предусмотреть возможность обращения к отдельным строкам массива по индексам, контроль выхода за пределы индексов, выполнения операций поэлементного сцепления двух массивов с образованием нового массива, слияния двух массивов с исключением повторяющихся элементов, печать (вывод на экран) элементов массива и всего массива

2. Описать класс «домашняя библиотека». Предусмотреть возможность работы с произвольным числом книг, поиска книги по какому-либо признаку (например, по автору или по году издания), добавления книг в библиотеку, удаления книг из нее, сортировки книг по разным полям. Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 10

1. Построить описание класса, содержащего информацию о почтовом адресе организации. Предусмотреть возможность раздельного изменения составных частей адреса, создания и уничтожения объектов этого класса.

2. Описать класс «записная книжка». Предусмотреть возможность работы с произвольным числом записей, поиска записи по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона), добавления и удаления записей, сортировки по разным полям. Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 11

1. Составить описание класса для представления комплексных чисел с возможностью задания вещественной и мнимой частей как числами типов double, так и целыми числами. Обеспечить выполнение операций сложения, вычитания и умножения комплексных чисел.

2. Описать класс «студенческая группа». Предусмотреть возможность работы с переменным числом студентов, поиска студента по какому-либо признаку (например, по фамилии, дате рождения или номеру телефона), добавления и удаления записей, сортировки по разным полям. Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 12

1. Составить описание класса для работы с цепными списками строк (строки произвольной длины) с операциями включения в список, удаления из списка элемента с заданным значением данного, удаления всего списка или конца списка, начиная с заданного элемента.

2. Описать класс, реализующий тип данных «вещественная матрица» и работу с ними. Класс должен реализовывать следующие операции над матрицами:

- сложение, вычитание, умножение, деление (+, -, *, /) (умножение и деление, как на другую матрицу, так и на число);
- операции сравнения на равенство/неравенство;
- операции вычисления обратной и транспонированной матрицы, операцию возведения в степень;
- методы, реализующие проверку типа матрицы (квадратная, диагональная, нулевая, единичная, симметрическая, верхняя треугольная, нижняя треугольная);

Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 13

1. Составить описание класса для объектов – векторов, задаваемых координатами концов в трехмерном пространстве. Обеспечить операции сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между векторами.

2. Описать класс «множество», позволяющий выполнять основные операции — добавление и удаление элемента, пересечение, объединение и разность множеств. Написать программу, демонстрирующую работу с этим классом. Программа должна содержать меню, позволяющее осуществить проверку всех методов класса.

Вариант 14

1. Составить описание класса прямоугольников со сторонами, параллельными осям координат. Предусмотреть возможность перемещения прямоугольников на плоскости, изменения размеров, построения наименьшего прямоугольника, содержащего два заданных прямоугольника, и прямоугольника, являющегося общей частью (пересечением) двух прямоугольников

2. Создать класс Money для работы с денежными суммами. Число должно быть представлено двумя полями: типа long для рублей и типа unsigned char — для копеек. Дробная часть (копейки) при выводе на экран должна быть отделена от целой части запятой. Реализовать сложение, вычитание, деление сумм, деление суммы на дробное число, умножение на дробное число и операции сравнения.

Вариант 15

1. Составить описание класса одномерных массивов строк, каждая строка которых задается длиной и указателем на выделенную для нее память. Предусмотреть возможность обращения к отдельным строкам массива по индексам, контроль выхода за пределы индексов, выполнения операций поэлементного сцепления двух массивов с образованием нового массива, слияния двух

массивов с исключением повторяющихся элементов, печать (вывод на экран) элементов массива и всего массива.

2. Создать класс Triangle для представления треугольника. Поля данных должны включать углы и стороны. Требуется реализовать операции: получения и изменения полей данных, вычисления площади, вычисления периметра, вычисления высот, а также определения вида треугольника (равносторонний, равнобедренный или прямоугольный).