ЛАБОРАТОРНАЯ РАБОТА 4. КЛАССЫ. ОБЪЕКТНОЕ МОДЕЛИРОВАНИЕ

1. Цель и содержание

Цель лабораторной работы: изучить структуру и принципы объявления классов, освоить технологию создания экземпляров классов (объектов).

Задачи лабораторной работы:

– научиться объявлять классы;
– научиться создавать объекты классов;
– научиться работать с полями данных и методами классов.

2. Теоретическая часть

2.1 Классы и структуры

Класс – это тип данных, объединяющий данные и методы их обработки. Класс – это пользовательский шаблон, в соответствии с которым можно создавать объекты. То есть класс – это правило, по которому будет строиться объект. Сам класс не содержит данных.

Объект класса (экземпляр класса) – переменная типа класс. Объект содержит данные и методы, манипулирующие этими данными. Класс определяет, какие данные содержит объект и каким образом он ими манипулирует.

Все, что справедливо для классов, можно распространить и на структуры. Отличие состоит в методе хранения объектов данных типов в оперативной памяти:структуры – это типы по значению, они размещаются в стеке; классы – это ссылочные типы, объекты классов размещаются в куче. Структуры не поддерживают наследование.

Структуры применяются для представления небольших объемов данных. Объявление структур происходит с использованием ключевого слова struct, объявление классов – с помощью ключевого слова class.

Пример объявления класса:

```
// Объявление класса
public class MyFirstClass
{
    // Данные-члены класса

    // Доступные на уровне экземпляра
    public int a;
    public float b__;
    public string fio;

    // Доступные только на уровне класса
    private bool IsOK;
    private double precision;
}
```

При создании как классов, так и структур, используется ключевое

слово new,например:

```
MyFirstClass obj = new MyFirstClass();
```

2.2 Структура класса

Данные и функции, объявленные внутри класса, называются членами класса(class members). Доступность членов класса может быть описана как public, private,protected, internal или internal protected.

2.2.1 Данные-члены

Данные-члены – это те структуры внутри класса, которые содержат данные класса – поля, константы события.

Поля – это любые переменные, ассоциированные с классом. После созданияэкземпляра класса к полям можно обращаться с использованием синтаксиса ,например: ИмяПоля ИмяОбъекта.

```
MyFirstClass obj = new MyFirstClass();

obj.a = 5;
int cc = obj.a;

obj.fio = "Novak E.I.";
```

Аналогичным образом с классом ассоциируются константы. События будут рассмотрены в следующих лабораторных работах.

2.2.2 Функции-члены

Функции-члены – это члены, которые обеспечивают некоторую функциональность для манипулирования данными классов. Они делятся на следующие виды: методы, свойства, конструкторы, финализаторы, операции и индексаторы.

Методы (method) – это функции, ассоциированные с определенным классом. Как и данные-члены, по умолчанию они являются членами экземпляра. Они могут быть объявлены статическими с помощью модификатора static.

Свойства (property) – это наборы функций, которые могут быть доступны клиенту таким же способом, как общедоступные поля класса. В C# предусмотрен специальный синтаксис для реализации чтения и записи свойств для классов, поэтому писать собственные методы с именами, начинающимися на Set и Get, не понадобится. Поскольку не существует какого-то отдельного синтаксиса для свойств, который отличал бы их от нормальных функций, создается иллюзия объектов как реальных сущностей, предоставляемых клиентскому коду.

Конструкторы (constructor) – это специальные функции, вызываемые автоматически при инициализации объекта. Их имена совпадают с именами классов, которым они принадлежат, и они не имеют типа возврата. Конструкторы полезны для инициализации полей класса.

Финализаторы (finalizer) похожи на конструкторы, но вызываются, когда среда CLR определяет, что объект больше не нужен. Они имеют то

же имя, что и класс, но с предшествующим символом тильды (~). Предсказать точно, когда будетвызван финализатор, невозможно.

Операции (operator) – это простейшие действия вроде + или -. Когда вы складываете два целых числа, то, строго говоря, применяете операцию + к целым. Однако C# позволяет указать, как существующие операции будут работать с пользовательскими классами (так называемая перегрузка операций).

Индексаторы (indexer) позволяют индексировать объекты таким же способом, как массив или коллекцию.

В данной лабораторной работе рассматриваются только методы класса – этофункции, ассоциированные с определенным классом.

В C# объявление метода класса состоит из спецификатора доступности,возвращаемого значения, имени метода, списка формальных параметров и тела метода:

```
[модификатор] тип_возврата имя_метода ([список_параметров])
{
    // тело метода
}
```

Например, добавим методы для объявленного ранее класса MyFirstClass:

```
// Объявление класса
public class MyFirstClass
{
    // Данные-члены класса

    // Доступные на уровне экземпляра
    public int a;
    public float b__;
    public string fio;

    // Доступные только на уровне класса
    private bool IsOK;
    private double precision;

    // Метод для инициализации некоторых полей класса
    public void InitClassMembers(int pA, float pB__, string pFio)
    {
        a = pA;
        b__ = pB__;
        fio = pFio;
    }

    // Метод, возвращающий значение вычисленное на основе полей класса
    public int GetAbsA()
    {
        return Math.Abs(a);
    }
}
```

Синтаксис вызова методов аналогичен синтаксису обращения к данным- членам:

```
MyFirstClass obj = new MyFirstClass();

obj.InitClassMembers(10, 0.8F, "Новиков П.Е.");

int abs_a = obj.GetAbsA();
```

В данном примере метод InitClassMembers не возвращает никаких данных, но требует передачи ему фактических параметров. В свою очередь, метод GetAbsAвозвращает значение типа int и не предполагает никаких параметров.

В общем случае параметры могут передаваться методу либо по значению, либо по ссылке. Когда переменная передается по ссылке, вызываемый метод получает саму переменную, поэтому любые изменения, которым она подвергнетсявнутри метода, останутся в силе после его завершения. Но если переменная передается по значению, вызываемый метод получает копию этой переменной, а это значит, что все изменения в ней по завершении метода будут утеряны. Для сложных типов данных передача по ссылке более эффективна из-за большого объема данных, который приходится копировать при передаче по значению.

Если не указано обратное, то в C# все параметры передаются по значению. Тем не менее, можно принудительно передавать значения по ссылке, для чего используется ключевое слово ref. Если параметр передается в метод, и входной аргумент этого метода снабжен префиксом ref, то любые изменения этой переменной, которые сделает метод, отразятся на исходном объекте.

В C-подобных языках функции часто возвращают более одного значения. Это обеспечивается применением выходных параметров, за счет присваивания значений переменным, переданным в метод по ссылке. Часто первоначальноезначение таких переменных не важно. Эти значения перезаписываются в функции,которая может даже не обращать внимания на то, что в них хранилось первоначально.

Было бы удобно использовать то же соглашение в C#. Однако в C# требуется, чтобы переменные были инициализированы каким-то начальным значением передтем, как к ним будет выполнено обращение. Хотя можно инициализировать входные переменные какими-то бессмысленными значениями до передачи их в функцию, которая наполнит их осмысленными значениями, этот прием выглядит влучшем случае излишним, а в худшем – сбивающим с толку. Тем не менее, существует способ обойти требование компилятора C# относительно начальной инициализации переменных.

Это достигается ключевым словом out. Когда входной аргумент снабжен префиксом out, этому методу можно передать неинициализированную переменную. Переменная передается по ссылке, поэтому любые изменения, выполненные методом в переменной, сохраняются после того, как он вернет управление. Ключевое слово out

также должно указываться при вызове метода – так же, как при его определении.

Частичные классы.

Ключевое слово partial (частичный) позволяет определить класс, структуру или интерфейс, распределенный по нескольким файлам. Но ситуации, когда множеству разработчиков требуется доступ к одному и тому же классу, или же в ситуации, когда некоторый генератор кода генерирует часть класса, такое разделение класса на несколько файлов может оказаться полезным. Ключевое слово partial просто помещается перед классом, структурой или интерфейсом.

Статические классы.

Статический класс функционально представляет собой то же самое, что и класс с приватным статическим конструктором. Создать экземпляр такого класса невозможно. Если указать ключевое слово static в объявлении класса, компилятор будет гарантировать, что к этому классу никогда не будут добавлены нестатические члены.

Класс Object.

Классы .NET изначально унаследованы от System.Object. Фактически, если при определении нового класса базовый класс не указан, компилятор автоматически предполагает, что он наследуется от Object.

Практическое значение этого в том, что помимо методов и свойств, которые программист определяет самостоятельно, также появляется доступ к множеству общедоступных и защищенных методов-членов, которые определены в классеObject. Эти методы присутствуют во всех определяемых классах.

3. Методика и порядок выполнения работы

1. Создайте консольное приложение в среде Visual Studio.
2. Изучите пример выполнения задания, представленный в данном разделе.
3. Выполните индивидуальное задание. Задания ориентированы на работу с классами.

Пример выполнения задания

Разработать класс для представления объекта «Прямоугольный параллелепипед». Реализуйте все необходимые поля данных (закрытые) и методы, позволяющие:

- устанавливать и считывать значения полей данных;
- вычислять объем прямоугольного параллелепипеда;
- вычислять площадь поверхности прямоугольного параллелепипеда;
- выводить полную информацию об объекте в консоль.

Решение данной задачи состоит из двух этапов: объявление класса Parallelepiped и демонстрация использования объекта данного класса.

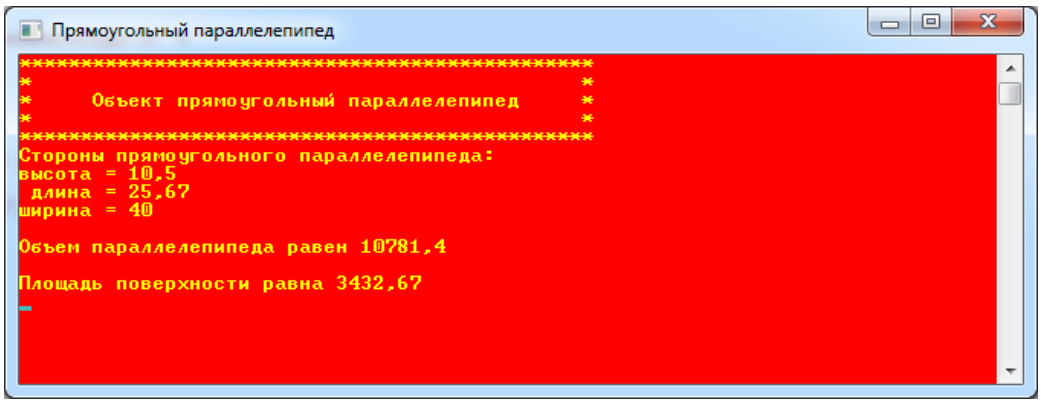
Полный листинг примера:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;

namespace LR_Three
{
    class Parallelepiped
    {
        // Поля данных представляют стороны параллелепипеда
        private double a, b, c;

        // Методы для установки и считывания значений полей
        public void Set_a(double pa) { a = pa; }
        public double Get_a() { return a; }

        public void Set_b(double pb) { b = pb; }
        public double Get_b() { return b; }

        public void Set_c(double pc) { c = pc; }
        public double Get_c() { return c; }

        // Метод для вычисления объема прямоугольного параллелепипеда
        public double GetV()
        {
            return a * b * c;
        }

        // Метод для вычисления площади поверхности прямоугольного параллелепипеда
        public double GetS()
        {
            return 2 * (a * b + b * c + a * c);
        }
```

```
        // Метод для вывода полной информации об объектк в консоль
        public void PrintFullInformation()
        {
            string str =    "*********************************************\n" +
                            "*                                           *\n" +
                            "*     Объект прямоугольный параллелепипед   *\n" +
                            "*                                           *\n" +
                            "*********************************************";
            Console.WriteLine(str);

            Console.WriteLine("Стороны прямоугольного параллелепипеда:\n" +
                "высота = {0}\n "+
                "длина = {1}"+
                "ширина = {2}", a, b, c);

            Console.WriteLine("Объем параллелепипеда равен {0}", GetV());

            Console.Write("Площадь поверхности равна {0}", GetS());
        }
    }

    class Program
    {
        static void Main(string[] args)
        {
            Console.Title = "Прямоугольный параллелепипед";
            Console.ForegroundColor = ConsoleColor.Yellow;
            Console.BackgroundColor = ConsoleColor.Red;
            Console.Clear();

            Parallelepiped p;

            p = new Parallelepiped();
            p.Set_a(10.5);
            p.Set_b(25.67);
            p.Set_c(40);

            p.PrintFullInformation();

            Console.ReadKey();
        }
    }


}
```

В данном примере необходимо обратит внимание на тот факт, что все вычисления выполняются внутри класса. Метод Main содержит только вызовы методов класса, то есть вся реализация скрыта.

В результате выполнения программы отобразится следующее консольное окно с выводом информации:



Индивидуальное задание 1

Спроектируйте класс, наполните его требуемой функциональностью,продемонстрируйте работоспособность класса.

Вариант	Выражение для вычисления
1.	Класс «Шар». Реализовать ввод и вывод полей данных, вычислениеобъема, диаметра и площади поверхности, а также вывод информации об объекте.
2.	Класс «Куб». Реализовать ввод и вывод полей данных, вычислениеобъема, площади поверхности, длины диагонали, а также вывод информации об объекте.
3.	Класс «Сфера». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а такжевывод информации об объекте.
4.	Класс «Точка в пространстве». Реализовать ввод и вывод полей данных, вычисление расстояния до введенной пользователем точки, расстояния от начала координат, а также вывод информации об объекте.
5.	Класс «График y=x». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем),длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте.
6.	Класс «Матрица MxN». Реализовать инициализацию элементов матрицы случайными числами, вывод матрицы, нахождение максимального и минимального элементов, а также вывод информации об объекте.
7.	Класс «Прямоугольный треугольник». Реализовать ввод и вывод полей данных, вычисление гипотенузы, площади и периметра, атакже вывод информации об объекте.
8.	Класс «Отрезок». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, атакже вывод информации об объекте.

9.	Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте.
10.	Класс «Ромб». Реализовать ввод и вывод полей данных (диагоналиромба), вычисление площади, периметра, а также вывод информации об объекте.
11.	Класс «График y=3x+5». Реализовать ввод и вывод полей данных,вычисление интеграла функции от a до b (вводятся пользователем), длины отрезка функции от (a, y(a)) до (b, y(b)), а также выводинформации об объекте.
12.	Класс «Матрица MxN». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение среднего арифметического всех элементов, а также вывод информации об объекте.
13.	Класс «График y=x-10». Реализовать ввод и вывод полей данных, вычисление интеграла функции от a до b (вводятся пользователем),длины отрезка функции от (a, y(a)) до (b, y(b)), а также вывод информации об объекте.
14.	Класс «Матрица MxN». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение и вывод среднего арифметического элементов в каждом столбце.

Индивидуальное задание 2

Спроектируйте класс, наполните его требуемой функциональностью, продемонстрируйте работоспособность класса.

Вариант 1

Создать класс Point3D, содержащий следующие члены класса:

– поля: int x, y, z;

– конструкторы, позволяющие создать экземпляр класса с нулевыми координатами, с заданными координатами;

– методы, позволяющие вывести координаты точки на экран, рассчитать расстояние от начала координат до точки, переместить точку на плоскости на вектор (a, b,c);

– свойства, позволяющие получить/установить координаты точки (доступное для чтения и записи), умножить координаты точки на скаляр (доступное только для записи);

– индексатор, позволяющий по индексу 0 обращаться к полю x, по индексу 1 –к полю y, 2- к полю z; при других значениях индекса выдается сообщение об ошибке;

– перегрузку:

- операции ++ (--) – одновременно увеличивает (уменьшает) значение полей на 1;

--констант true и false – обращение к экземпляру класса дает значение true,

если значение полей x и y совпадает, иначе false;

- операции бинарный + – одновременно добавляет к полям значение скаляра.

Вариант 2

Создать класс Triangle, содержащий следующие члены класса:

– поля: int a, b, c;

– конструктор, позволяющий создать экземпляр класса с заданными длинами сторон;

– методы, позволяющие вывести длины сторон треугольника на экран, рассчитать периметр треугольника, рассчитать площадь треугольника;

– свойства, позволяющие получить/установить длины сторон треугольника (доступное для чтения и записи), установить, существует ли треугольник с данными длинами сторон (доступное только для чтения);

– индексатор, позволяющий по индексу 1 обращаться к полю a, по индексу 2 –к полю b, по индексу 3– к полю c, при других значениях индекса выдается сообщение об ошибке;

– перегрузку:

- операции ++ (--) – одновременно увеличивает (уменьшает) значение полей a, b и c на 1;

– констант true и false – обращение к экземпляру класса дает значение true, если треугольник с заданными длинами сторон существует, иначе – false;

– операции * – одновременно умножает поля a, b и c на скаляр.

Вариант 3

Создать класс Rectangle, содержащий следующие члены класса:
– поля: int a, b;

– конструктор, позволяющий создать экземпляр класса с заданными длинами сторон;

– методы, позволяющие вывести длины сторон прямоугольника на экран, рассчитать периметр прямоугольника, рассчитать площадь прямоугольника;

– свойства, позволяющие получить/установить длины сторон прямоугольника (доступное для чтения и записи), установить, является ли данный прямоугольник квадратом (доступное только для чтения);

– индексатор, позволяющий по индексу 1 обращаться к полю a, по индексу 2 – к полю b, при других значениях индекса выдается сообщение об ошибке;

– перегрузку:

- операции ++ (--) – одновременно увеличивает (уменьшает) значение полей a и b на 1;

- константы true и false – обращение к экземпляру класса дает значение true, если прямоугольник с заданными длинами сторон является квадратом, иначе – false;

- операции * – одновременно умножает поля a и b на скаляр.

Вариант 4

Создать класс Ellipse, содержащий следующие члены класса:
– поля: int a, b;

– конструктор, позволяющий создать экземпляр класса с заданными длинами полуосей;

– методы, позволяющие вывести длины полуосей эллипса на экран, рассчитать периметр, рассчитать площадь эллипса;

– свойства, позволяющие получить/установить длины полуосей (доступное для чтения и записи), установить, является ли данный эллипс окружностью (доступное только для чтения);

– индексатор, позволяющий по индексу 1 обращаться к полю a, по индексу 2 – к полю b, при других значениях индекса выдается сообщение об ошибке;

– перегрузку:

-операции ++ (--) – одновременно увеличивает (уменьшает) значение полей a и b на 1;

- константа true и false – обращение к экземпляру класса дает значение true, если эллипс с заданными длинами полуосей является окружностью, иначе – false;

- операции * – одновременно умножает поля a и b на скаляр.

Вариант 5

Создать класс Money, содержащий следующие члены класса:

– поля: int nominal (номинал купюры); int quantity (количество купюр);

– конструктор, позволяющий создать экземпляр класса с заданными значениям полей;

– методы, позволяющие вывести номинал и количество купюр, определить, хватит ли денежных средств на покупку товара на сумму N рублей, определить, сколько штук товара стоимости n рублей можно купить на имеющиеся денежные средства;

– свойства, позволяющие получить/установить значение полей (доступное для чтения и записи), рассчитать сумму денег (доступное только для чтения);

– индексатор, позволяющий по индексу 1 обращаться к полю nominal, по индексу 2 – к полю quantity, при других значениях индекса выдается сообщение об ошибке;

– перегрузку:

-операции ++ (--) – одновременно увеличивает (уменьшает) значение полей;

- операции ! – возвращает значение true, если поле quantity не нулевое, иначе false;

- операции бинарный + – добавляет к значению поля quantity значение скаляра.

Вариант 6

Создать класс для работы с одномерным массивом целых чисел. Разработать следующие члены класса:

– поля: int [] IntArray, int n;

– конструктор, позволяющий создать массив размерности n;

– методы, позволяющие ввести элементы массива с клавиатуры, вывести элементы массива на экран, отсортировать элементы массива в порядке возрастания;

– свойство, возвращающее размерность массива (доступное только для чтения), и свойство, позволяющее умножить все элементы массива на скаляр (доступное только для записи);

– индексатор, позволяющий по индексу обращаться к соответствующему элементу массива;

– перегрузку:

-операции ++ (--) – одновременно увеличивает (уменьшает) значение всех элементов массива на 1;

- операции ! – возвращает значение true, если элементы массива не упорядочены по возрастанию, иначе – false;

-операции бинарный * – умножить все элементы массива на скаляр;

-операции преобразования класса массив в одномерный массив (и наоборот).

Вариант 7

Создать класс для работы с двумерным массивом целых чисел. Разработать следующие члены класса:

– поля: int [,] intArray, int n, int m;

– конструктор, позволяющий создать массив размерности n×m;

– методы, позволяющие ввести элементы массива с клавиатуры, вывести элементы массива на экран, вычислить сумму элементов i-го столбца;

– свойства, позволяющие вычислить количество нулевых элементов в массиве (доступное только для чтения), установить значение всех элементов главной диагонали массива, равное скаляру (доступное только для записи);

– двумерный индексатор, позволяющий обращаться к соответствующему элементу массива;

– перегрузку:

- операции ++ (--) – одновременно увеличивает (уменьшает) значение всех элементов массива на 1;
- констант true и false – обращение к экземпляру класса дает значение true, если двумерный массив является квадратным;
- операции бинарный + – сложить два массива соответствующих размерностей;
- операции преобразования класса массив в двумерный массив (и наоборот).

Вариант 8

Создать класс для работы с двумерным массивом вещественных чисел. Разработать следующие функциональные члены класса:

– поля: double [][] doubelArray;

– конструктор, позволяющий создать ступенчатый массив;

– методы, позволяющие ввести элементы массива с клавиатуры, вывести элементы массива на экран, отсортировать элементы каждой строки массива в порядке убывания;

– свойство, возвращающее общее количество элементов в массиве (доступное только для чтения), и свойство, позволяющее увеличить значение всех элементов массива на скаляр (доступное только для записи);

– двумерный индексатор, позволяющий обращаться к соответствующему элементу массива;

– перегрузку:

- операции ++ (--) – одновременно увеличивает (уменьшает) значение всех элементов массива на 1;
- константа true и false – обращение к экземпляру класса дает значение true, если каждая строка массива упорядочена по возрастанию, иначе – false;
- операции преобразования класса массив в ступенчатый массив (и наоборот).

Вариант 9

Создать класс для работы со строками. Разработать следующие члены класса:

– поле: string line;

– конструктор, позволяющий создать строку на основе заданного строкового литерала;

– методы, позволяющие подсчитать количество цифр в строке, выводить на экран все символы строки, встречающиеся в ней ровно один раз, вывести на экран самую длинную последовательность повторяющихся символов в строке;

– свойство, возвращающее общее количество символов в строке (доступное только для чтения);

– индексатор, позволяющий по индексу обращаться к соответствующему символу строки (доступный только для чтения);

– перегрузку:

- операции унарного ! – возвращает значение true, если строка не пустая, иначе – false;
- константа true и false – обращение к экземпляру класса дает значение true, если строка является палиндромом, false – в противном случае;
- операции & – возвращает значение true, если строковые поля двух объектов посимвольно равны (без учета регистра), иначе – false;
- операции преобразования класса строка в тип string (и наоборот).

Вариант 10

Создать класс для работы со строками. Разработать следующие члены класса:

– поле: StringBuilder line;

– конструктор, позволяющий создать строку на основе заданного строкового литерала, и конструктор, позволяющий создавать пустую строку;

– методы, позволяющие подсчитать количество пробелов в строке, заменить в строке все прописные символы на строчные, удалить из строки все знаки препинания;

– свойство, возвращающее общее количество элементов в строке (доступное только для чтения), и свойство, позволяющее установить значение поля в соответствии с введенным значением строки с клавиатуры, а также получить значение данного поля (доступно для чтения и записи);

– индексатор, позволяющий по индексу обращаться к соответствующему символу строки;

– перегрузку:

- операции унарного + (-) – преобразующей строку к строчным (прописным) символам;
- констант true и false – обращение к экземпляру класса дает значение true, если строка не пустая, иначе – false;
- операции & – возвращает значение true, если строковые поля двух объектов посимвольно равны (без учета регистра), иначе – false;
- операции преобразования строки в тип StringBuilder (и наоборот).

Вариант 11

Самостоятельно изучите тип данных DateTime, на основе которого необходимо создать класс для работы с датой. Данный класс должен содержать следующие члены класса:

– поле DataTime data;

– конструкторы, позволяющие установить заданную дату, дату 1.01.2000;

– методы, позволяющие вычислить дату предыдущего дня, вычислить дату следующего дня, определить сколько дней осталось до конца месяца;

– свойства, позволяющие установить или получить значение поле класса (доступно для чтения и записи), определить, является ли год високосным (доступно только для чтения);

– индексатор, позволяющий определить дату i-го по счету дня относительно установленной даты (при отрицательных значениях индекса отсчет ведется в обратном порядке);

– перегрузку: операции ! – возвращает значение true, если установленная дата не является последним днем месяца, иначе – false;

– констант true и false – обращение к экземпляру класса дает значение true, если установленная дата является началом года, иначе – false;

– операции & – возвращает значение true, если поля двух объектов равны, иначе false.

Вариант 12

Создать класс GDS, содержащий следующие члены класса:

– поля: string name (название товара); int quantity (количество товара); price (цена);

– конструктор, позволяющий создать экземпляр класса с заданными значениям полей;

– методы, позволяющие вывести название товара, определить, сколько единиц товара можно купить на сумму N рублей, определить, сколько необходимо денег для покупки n единиц товара;

– свойства, позволяющие получить/установить значение полей (доступное для чтения и записи), рассчитать сумму денег (доступное только для чтения);

– индексатор, позволяющий по индексу 1 обращаться к полю name, по индексу 2 – к полю quantity, по индексу 3 – к полю price; при других значениях индекса выдается сообщение об ошибке;

– перегрузку:

-операции ++ (--) – одновременно увеличивает (уменьшает) значение поля quantity;

- операции ! – возвращает значение true, если поле price не нулевое, иначе false;

- операции бинарный + – добавляет к значению поля quantity значение скаляра.

Содержание отчета и его форма

Отчет по лабораторной работе должен содержать:

1. Номер и название лабораторной работы.
2. Цели лабораторной работы.
3. Ответы на контрольные вопросы.
4. Экранные формы и листинг программного кода, показывающие порядок выполнения лабораторной работы, и результаты, полученные в ходе её выполнения.

Отчет о выполнении лабораторной работы в электронном виде прикрепляется на портал.