# ЛАБОРАТОРНАЯ РАБОТА 8. КЛАССЫ. СТРУКТУРЫ.

## 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. Формируемые компетенции

Лабораторная работа направлена на формирование следующих компетенций:

– способность к проектированию базовых и прикладных информационных технологий (ПК-11);

– способность разрабатывать средства реализации информационных технологий (методические, информационные, математические, алгоритмические, технические и программные) (ПК-12).

## 3. Теоретическая часть

### 3.1 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();
```

3.2 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.

3.2.1 Данные-члены.

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

Поля – это любые переменные, ассоциированные с классом. После создания экземпляра 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.

3.2.2 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 будет вызван 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 – это функции, 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 GetAbsA возвращает 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 подвергнется внутри 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 первоначальное значение 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 перед тем, 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 в лучшем 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.

Rлассы .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 классе Object. Эти методы присутствуют во всех определяемых классах.

## 4. Оборудование и материалы

Для выполнения лабораторной работы рекомендуется использовать персональный компьютер со следующими характеристиками: 64-разрядный (x64) процессор с тактовой частотой 1 ГГц и выше, оперативная память – 1 Гб и выше, свободное дисковое пространство – не менее 1 Гб, графическое

устройство DirectX 9. Программное обеспечение: операционная система WINDOWS 7 и выше, Microsoft Visual Studio 20112 и выше.

5. Указания по технике безопасности

Техника безопасности при выполнении лабораторной работы определяется общепринятой для пользователей персональных компьютеров. Самостоятельно не производить ремонт персонального компьютера, установку и удаление программного обеспечения; в случае неисправности персонального компьютера сообщить об этом обслуживающему персоналу лаборатории; не касаться электрических розеток металлическими предметами; рабочее место пользователя персонального компьютера должно содержаться в чистоте; не разрешается возле персонального компьютера принимать пищу, напитки.

6. Методика и порядок выполнения работы

1. Создайте консольное приложение в соответствии с алгоритмом, представленным в лабораторной работе №1.

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.

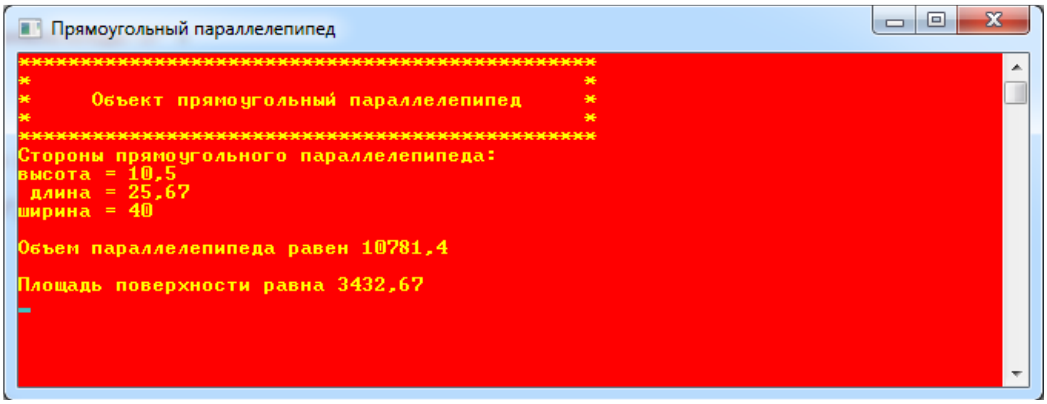
Полный листинг примера:

```
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.**

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

| Вариант | Выражение для вычисления |
|---|---|
| **1.** | Класс «Шар». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте. |
| **2.** | Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, 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 также вывод 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 «Шар». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте. |
| 7. | Класс «Матрица MxN». Реализовать инициализацию элементов матрицы случайными числами, вывод матрицы, нахождение максимального и минимального элементов, а также вывод информации об объекте. |
| **8.** | Класс «Прямоугольный треугольник». Реализовать ввод и вывод полей данных, вычисление гипотенузы, площади и периметра, а также вывод информации об объекте. |

| | |
|---|---|
| **9.** | Класс «Отрезок». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте. |
| **10.** | Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте. |
| **11.** | Класс «Ромб». Реализовать ввод и вывод полей данных (диагонали ромба), вычисление площади, периметра, а также вывод информации об объекте. |
| **12.** | 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. |
| **13.** | 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 вывод информации об объекте. |
| **14.** | Класс «Матрица MxN». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение среднего арифметического всех элементов, а также вывод информации об объекте. |
| **15.** | Класс «Отрезок в пространстве». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте. |
| **16.** | 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. |
| **17.** | Класс «Матрица MxN». Реализовать инициализацию элементов матрицы случайными числами, вывод транспонированной матрицы, нахождение и вывод среднего арифметического элементов в каждом столбце. |
| **18.** | Класс «Куб». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, длины диагонали, а также вывод информации об объекте. |
| **19.** | Класс «Сфера». Реализовать ввод и вывод полей данных, вычисление объема, диаметра и площади поверхности, а также вывод информации об объекте. |
| **20.** | 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. |

| | |
|---|---|
| **21.** | 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. |
| **22.** | 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. |
| **23.** | 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. |
| **24.** | Класс «Цилиндр». Реализовать ввод и вывод полей данных, вычисление объема, площади поверхности, а также вывод информации об объекте. |
| **25.** | Класс «Отрезок в пространстве». Реализовать ввод и вывод полей данных (координаты начала и координаты конца отрезка), вычисление длины, расстояний начала и конца отрезка от начала координат, а также вывод информации об объекте. |

## 7. 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 письменном виде сдается преподавателю.

## 8. Контрольные вопросы

1. Что такое класс?

2. Что такое структура? Чем структура отличается от класса?

3. Что такое члены класса? Какие группы членов класса вы знаете?

4. Какие типы членов-данных вы знаете?

5. Какие типы функций-членов класса вы знаете?

6. Как поменять цвет текста в консольном приложении?

7. Как поменять цвет фона в консольном приложении?

8. Какие модификаторы доступности членов класса вы знаете?

9. Какое ключевое слово используется для создания объекта класса?

## 9. Список литературы

Для выполнения лабораторной работы, при подготовке к защите, а также для ответа на контрольные вопросы рекомендуется использовать следущие источники: [8].

# ЛАБОРАТОРНАЯ РАБОТА 9. КОНСТРУКТОР КЛАССА. ПЕРЕГРУЗКА КОНСТРУКТОРОВ КЛАССА.

## 1. Цель и содержание

Цель лабораторной работы: понять принципы работы конструктора.

Задачи лабораторной работы:

– научиться объявлять конструктор класса;

– научиться создавать перегруженные конструкторы.

## 2. Формируемые компетенции

Лабораторная работа направлена на формирование следующих компетенций:

– способность к проектированию базовых и прикладных информационных технологий (ПК-11);

– способность разрабатывать средства реализации информационных технологий (методические, информационные, математические, алгоритмические, технические и программные) (ПК-12).

## 3. Теоретическая часть

Конструктор – это метод класса, который не возвращает значения и имеет то же самое имя, что и класс. Если конструктор класса не определен программистом явно, то компилятор создаст конструктор по умолчанию.

Конструкторы подчиняются тем же правилам перегрузки, что и все методы.

В C# поддерживается перегрузка методов – то есть может существовать несколько версий одного метода, но с разными сигнатурами (методы