Лабораторная работа № 1.  
Введение в классы

**Цель лабораторной работы**

Закрепление теоретических знаний по созданию классов.

**Постановка задачи**

Класс является основой для создания объектов. В классе определяются данные и код, который работает с этими данными. Объекты являются экземплярами класса. Методы и переменные, составляющие класс, называются членами класса. При определении класса объявляются данные, которые он содержит, и код, работающий с этими данными. Данные содержатся в переменных экземпляра, которые определены классом, а код содержится в методах. В С# определены несколько специфических разновидностей членов класса. Это — переменные экземпляра, статические переменные, константы, методы, конструкторы, деструкторы, индексаторы, события, операторы и свойства.

Непосредственно инициализация переменных в объекте (переменных экземпляра) происходит в конструкторе. В классе могут быть определены несколько конструкторов.

class название\_класса

{

// содержимое класса

}

class Person

{

}

**Создание объекта класса**

После определения класса мы можем создавать его объекты. Для создания объекта применяются **конструкторы**. По сути конструкторы представляют специальные методы, которые называются так же как и класс, и которые вызываются при создании нового объекта класса и выполняют инициализацию объекта.

Person tom = new Person(); // создание объекта класса Person

class Person // определение класса Person

{

}

Person tom = new Person(); // создание объекта класса Person

class Person // определение класса Person

{

public Person() //Конструктор

{

//Если конструктор пустой, то создавать его не обязательно

}

}

В строке Person tom = new Person(); Круглые скобки означают вызов конструктора.

Заметим, что для создания объекта всегда используется специальное слово new.

Чтобы конструктор был доступен вне класса Person используется модификатор public.

**Поля и методы класса**

Класс может хранить некоторые данные. Для хранения данных в классе применяются **поля**. По сути **поля класса** - это переменные, определенные на уровне класса.

Кроме того, класс может определять некоторое поведение или выполняемые действия. Для определения поведения в классе применяются методы.

class Person

{

public string name = "Undefined"; // имя

public int age; // возраст

public void Print()

{

Console.WriteLine($"Имя: {name} Возраст: {age}");

}

}

В отличие от переменных, определенных в методах, поля класса могут иметь **модификаторы**, которые указываются перед полем.

**Обращение к функциональности класса**

Для обращения к функциональности класса - полям, методам (а также другим элементам класса) применяется точечная нотация точки - после объекта класса ставится точка, а затем элемент класса:

using System;

Person tom = new Person(); // создание объекта класса Person

string personName = tom.name; // Получаем значение полей в переменные

int personAge = tom.age;

Console.WriteLine($"Имя: {personName} Возраст {personAge}");

// Имя: Undefined Возраст: 0

tom.name = "Tom"; // устанавливаем новые значения полей

tom.age = 37;

// обращаемся к методу Print

tom.Print(); // Имя: Tom Возраст: 37

class Person //Конструкторов может быть несколько

{

public string name;

public int age;

public Person()

{

name = "Неизвестно"; age = 18;

}

public Person(string n)

{

name = n; age = 18;

}

public Person(string n, int a)

{

name = n; age = a;

}

public void Print() =>

Console.WriteLine($"Имя: {name} Возраст: {age}");

}

**Цепочка вызова конструкторов**

В примере выше определены три конструктора. Все три конструктора выполняют однотипные действия - устанавливают значения полей name и age. Но этих повторяющихся действий могло быть больше. И мы можем не дублировать функциональность конструкторов, а просто обращаться из одного конструктора к другому также через ключевое слово **this**, передавая нужные значения для параметров:

class Person

{

public string name;

public int age;

public Person() : this("Неизвестно")

{

}

public Person(string name) : this(name, 18)

{

}

public Person(string name, int age)

{

this.name = name;

this.age = age;

}

public void Print() =>

Console.WriteLine($"Имя: {name} Воз-раст: {age}");

}

**Инициализаторы объектов**

Для инициализации объектов классов можно применять **инициализаторы**. Инициализаторы представляют передачу в фигурных скобках значений доступным полям и свойствам объекта:

Person tom = new Person { name = "Tom", age = 31 };

// или так

// Person tom = new() { name = "Tom", age = 31 };

tom.Print(); // Имя: Tom Возраст: 31

С помощью инициализатора мы можем установить значения **только доступных из вне** класса полей и свойств объекта. Например, в примере выше поля **name** и **age** имеют модификатор доступа public, поэтому они доступны из любой части программы.

Инициализатор выполняется после конструктора, поэтому если и в конструкторе, и в инициализаторе устанавливаются значения одних и тех же полей, и свойств, то значения, устанавливаемые в конструкторе, заменяются значениями из инициализатора.

Person tom = new Person { Name = "Tom", Company = { Title = "Microsoft" } };

tom.Print(); // Имя: Tom Компания: Microsoft

class Person

{

public string Name;

public Company Company;

public Person()

{

Name = "Undefined";

Company = new Company();

}

public void Print() => Console.WriteLine($"Имя: {Name} Компания: {Company.Title}");

}

class Company

{

public string Title;

}

**Задание на лабораторную работу**

1. Разработать консольное приложение и реализовать класс согласно заданию.
2. Продемонстрировать работу с классом: создать экземпляры класса, вызвать методы.
3. Предусмотреть защиту от создания класса с некорректными значениями полей, предусмотреть проверки данных в методах.
4. Аккуратно оформить программу.

**Варианты заданий на лабораторную работу**

1. Класс «QuadraticEquationSolver»

**Применение:** Решение квадратных уравнений в научных и инженерных расчётах  
**Поля:**

* a, b, c (коэффициенты уравнения ax² + bx + c = 0)
* Precision (точность вычислений)

**Конструктор:**  
QuadraticEquationSolver(double a, double b, double c)

**Методы:**

1. GetDiscriminant() → возвращает D = b² - 4ac
2. Solve() → возвращает массив корней
3. GetVertex() → координаты вершины параболы
4. HasRealRoots() → проверка существования действительных корней
5. Derivative() → производная уравнения (2ax + b)

2. Класс «Vector2D»

**Применение:** Работа с векторами в физике/графике  
**Поля:**

* X, Y
* Normalized (флаг единичного вектора)

**Конструктор:**  
Vector2D(double x, double y)

**Методы:**

1. Magnitude() → длина вектора: √(X² + Y²)
2. DotProduct(Vector2D other) → скалярное произведение
3. AngleBetween(Vector2D other) → угол между векторами
4. Rotate(double angle) → поворот на заданный угол
5. Project(Vector2D axis) → проекция на другой вектор

3. Класс «StatisticsCalculator»

**Применение:** Анализ данных в экономике/биологии  
**Поля:**

* Data (список чисел)
* IsSample (флаг выборки)

**Конструктор:**  
StatisticsCalculator(List<double> data)

**Методы:**

1. Mean() → среднее значение
2. Median() → медиана
3. StandardDeviation() → стандартное отклонение
4. Normalize() → нормализация данных (0–1)
5. Outliers() → список выбросов (правило 3σ)

4. Класс «MortgageCalculator»

**Применение:** Финансовые расчёты для кредитов  
**Поля:**

* Principal, AnnualRate, Years
* PaymentsPerYear

**Конструктор:**  
MortgageCalculator(double principal, double rate, int years)

**Методы:**

1. MonthlyPayment() → аннуитетный платёж
2. TotalInterest() → общая переплата
3. PaymentSchedule() → график платежей
4. EarlyRepaymentImpact(double extraPayment) → эффект досрочного погашения
5. InterestToPrincipalRatio() → соотношение процентов/основного долга

5. Класс «Matrix2x2»

**Применение:** Линейные преобразования в компьютерной графике  
**Поля:**

* M11, M12, M21, M22

**Конструктор:**  
Matrix2x2(double m11, double m12, double m21, double m22)

**Методы:**

1. Determinant() → M11\*M22 - M12\*M21
2. Inverse() → обратная матрица
3. Transpose() → транспонированная матрица
4. Eigenvalues() → собственные числа
5. TransformVector(Vector2D vec) → применение к вектору

6. Класс «ComplexNumber»

**Применение:** Электротехника, обработка сигналов  
**Поля:**

* Real, Imaginary

**Конструктор:**  
ComplexNumber(double re, double im)

**Методы:**

1. Add(ComplexNumber other) → сложение
2. PolarForm() → модуль и аргумент
3. Conjugate() → комплексно-сопряжённое
4. Multiply(ComplexNumber other) → умножение
5. Roots(int n) → корни n-й степени

7. Класс «GeometryHelper»

**Применение:** CAD-системы, игры  
**Статические методы:**

1. CircleArea(radius) → πr²
2. TriangleArea(base, height) → (base \* height)/2
3. PointDistance(x1, y1, x2, y2) → расстояние между точками
4. LineCircleIntersection(cx, cy, r, k, b) → точки пересечения прямой и окружности
5. IsPointInPolygon(polygon, point) → проверка принадлежности точки многоугольнику

8. Класс «PrimeNumberChecker»

**Применение:** Криптография  
**Поля:**

* Number
* CacheLimit

**Методы:**

1. IsPrime() → проверка простоты
2. GetPrimesInRange(start, end) → простые числа в диапазоне
3. Factorization() → разложение на множители
4. IsCoprime(num) → взаимно простые числа
5. NextPrime() → следующее простое число

9. Класс «Fraction»

**Применение:** Точные вычисления с дробями  
**Поля:**

* Numerator, Denominator

**Конструктор:**  
Fraction(int num, int denom)

**Методы:**

1. Add(Fraction other) → сложение
2. Simplify() → сокращение
3. ToDecimal() → десятичное представление
4. Reciprocal() → обратная дробь
5. CompareTo(Fraction other) → сравнение

10. Класс «Point3D»

**Применение:** 3D-моделирование  
**Поля:**

* X, Y, Z

**Конструктор:**  
Point3D(double x, double y, double z)

**Методы:**

1. DistanceTo(Point3D other) → расстояние
2. ProjectTo2D(plane) → проекция на плоскость
3. Mirror(plane) → отражение
4. IsCollinear(p1, p2) → проверка коллинеарности
5. ToSpherical() → сферические координаты

11. Класс «Polynomial»

**Применение:** Математический анализ  
**Поля:**

* Coefficients (список)

**Конструктор:**  
Polynomial(List<double> coeffs)

**Методы:**

1. Evaluate(x) → значение в точке
2. Derivative() → производная
3. Integral() → первообразная
4. FindRoots() → численное решение
5. Add(Polynomial other) → сложение многочленов

12. Класс «BMIcalculator»

**Применение:** Медицинские расчёты  
**Поля:**

* WeightKg, HeightCm

**Конструктор:**  
BMIcalculator(double weight, double height)

**Методы:**

1. Calculate() → индекс массы тела
2. GetCategory() → классификация
3. IdealWeight() → идеальный вес
4. BodySurfaceArea() → площадь тела
5. HealthRiskAssessment() → оценка рисков

13. Класс «Matrix3x3»

**Применение:** 3D-графика  
**Поля:**

* M11..M33 (9 элементов)

**Конструктор:**  
Matrix3x3(double[] elements)

**Методы:**

1. Determinant() → правило Саррюса
2. Multiply(Matrix3x3 other) → умножение
3. RotationMatrix(angle, axis) → матрица поворота
4. Inverse() → обратная матрица
5. TransformVector(Vector3D vec) → преобразование

14. Класс «Vector3D»

**Применение:** Физика движений  
**Поля:**

* X, Y, Z

**Конструктор:**  
Vector3D(double x, double y, double z)

**Методы:**

1. CrossProduct(Vector3D other) → векторное произведение
2. AngleBetween(Vector3D other) → угол
3. Magnitude() → длина
4. Normalize() → нормирование
5. Rotate(Quaternion q) → поворот

15. Класс «ProjectileMotion»

**Применение:** Баллистика  
**Поля:**

* InitialVelocity, Angle, Gravity

**Конструктор:**  
ProjectileMotion(double v0, double angle, double g)

**Методы:**

1. CalculateMaxHeight() → максимальная высота
2. CalculateRange() → дальность
3. TimeOfFlight() → время полёта
4. TrajectoryPoint(t) → координата в момент t
5. ImpactVelocity() → скорость при ударе

16. Класс «OhmsLaw»

**Применение:** Электротехника  
**Поля:**

* Voltage, Current, Resistance

**Конструктор:**  
OhmsLaw(double v, double i, double r)

**Методы:**

1. CalculateVoltage() → V = I\*R
2. CalculatePower() → P = I²\*R
3. SeriesResistance(r1, r2) → общее сопротивление
4. ParallelResistance(r1, r2) → для параллельных
5. VoltageDivider(r1, r2) → делитель напряжения

17. Класс «ThermalExpansion»

**Применение:** Термодинамика  
**Поля:**

* InitialLength, Coefficient, DeltaTemp

**Конструктор:**  
ThermalExpansion(double l0, double α, double ΔT)

**Методы:**

1. CalculateExpandedLength() → новая длина
2. ThermalStress(youngModulus) → напряжение
3. CriticalTemp() → температура разрушения
4. VolumeExpansion() → объёмное расширение
5. CompensationGap() → зазор для компенсации

18. Класс «LensFormula»

**Применение:** Оптика  
**Поля:**

* FocalLength, ObjectDistance, ImageDistance

**Конструктор:**  
LensFormula(double f, double d0, double di)

**Методы:**

1. CalculateImageDistance() → положение изображения
2. Magnification() → увеличение
3. LensPower() → оптическая сила
4. CombinedFocalLength(f1, f2) → для системы линз
5. ImageType() → действительное/мнимое

19. Класс «KineticEnergy»

**Применение:** Механика  
**Поля:**

* Mass, Velocity

**Конструктор:**  
KineticEnergy(double m, double v)

**Методы:**

1. CalculateEnergy() → E = mv²/2
2. Momentum() → импульс
3. VelocityFromEnergy(e) → скорость из энергии
4. RelativisticEnergy() → с учётом ОТО
5. WorkEnergyTheorem(force, distance) → работа

20. Класс «Quadrilateral»

**Применение:** Геометрия  
**Поля:**

* SideA, SideB, SideC, SideD, Angle

**Конструктор:**  
Quadrilateral(double a, double b, double c, double d, double angle)

**Методы:**

1. CalculateArea() → площадь
2. Perimeter() → периметр
3. DiagonalLengths() → длины диагоналей
4. IsCyclic() → можно ли вписать окружность
5. Type() → тип (ромб, трапеция и т.д.)

**Пример разработки приложения**

**Задание 1.**

Создайте класс **StudentNotebook**, который будет хранить информацию о студенте и его учебных достижениях. Класс должен содержать:

1. **Поля класса**:
   * studentName (имя студента)
   * studentAge (возраст)
   * subjects (список предметов)
   * grades (словарь: предмет → список оценок)
2. **Конструктор**:
   * Принимает имя и возраст
   * Инициализирует пустые списки предметов и оценок
3. **Методы**:
   * AddSubject() - добавляет новый предмет
   * AddGrade() - добавляет оценку по предмету
   * GetAverageGrade() - вычисляет средний балл
   * PrintInfo() - выводит информацию о студенте
   * PrintGrades() - показывает все оценки

**Шаг 1: Понимание задачи**

Требуется создать класс StudentNotebook, который будет хранить информацию о студенте (имя, возраст), его предметах и оценках по ним. Класс должен предоставлять методы для добавления предметов, добавления оценок, вычисления среднего балла, вывода информации о студенте и его оценках.

**Шаг 2: Определение структуры класса**

* **Поля**:
  + studentName (string) — имя студента.
  + studentAge (int) — возраст студента.
  + subjects (List<string>) — список предметов.
  + grades (Dictionary<string, List<int>>) — словарь, где ключ — предмет, а значение — список оценок по нему.
* **Конструктор**:
  + Принимает имя и возраст студента.
  + Инициализирует пустые списки subjects и grades.
* **Методы**:
  + AddSubject(string subject) — добавляет предмет в список.
  + AddGrade(string subject, int grade) — добавляет оценку по предмету.
  + GetAverageGrade() — вычисляет средний балл по всем предметам.
  + PrintInfo() — выводит имя и возраст студента.
  + PrintGrades() — выводит все оценки по предметам.

**Шаг 3: Реализация класса на C#**

public class StudentNotebook

{

// Поля класса

private string studentName;

private int studentAge;

private List<string> subjects;

private Dictionary<string, List<int>> grades;

// Конструктор

public StudentNotebook(string name, int age)

{

studentName = name;

studentAge = age;

subjects = new List<string>();

grades = new Dictionary<string, List<int>>();

}

// Метод для добавления предмета

public bool AddSubject(string subject)

{

if (subjects.Contains(subject))

return false;

subjects.Add(subject);

grades[subject] = new List<int>();

return true;

}

// Метод для добавления оценки по предмету

public bool AddGrade(string subject, int grade)

{

if (!subjects.Contains(subject))

return false;

grades[subject].Add(grade);

return true;

}

// Метод для вычисления среднего балла

public double GetAverageGrade()

{

if (grades.Count == 0)

return 0;

double sum = 0;

foreach (var (subject, subjectGrades) in grades)

{

sum += GetAverageGrade(subject);

}

return sum / grades.Count;

}

// Метод для вычисления среднего балла по предмету

public double GetAverageGrade(string subject)

{

if (grades.Count == 0)

return 0;

if (!grades.TryGetValue(subject, out var grages))

return 0;

return grages.Average();

}

// Метод для вывода информации о студенте

public void PrintInfo()

{

// Смешивание логики и I/O плохо, но мы оставим

Console.WriteLine($"Имя: {studentName}");

Console.WriteLine($"Возраст: {studentAge}");

}

// Метод для вывода всех оценок

public void PrintGrades()

{

// Смешивание логики и I/O плохо, но мы оставим

Console.WriteLine("Оценки:");

foreach (var subject in subjects)

{

Console.Write($"{subject}: ");

if (grades.ContainsKey(subject) && grades[subject].Count > 0)

{

Console.WriteLine(string.Join(", ", grades[subject]));

}

else

{

Console.WriteLine("нет оценок");

}

}

}

}

# Содержание пояснительной записки

1. Постановка задачи. Приводится теоретический материал, использованный при написании приложения.

2. Формулировка задания и вариант. Приводится задание на лабораторную работу и вариант этого задания.

3. Описание выполняемых действий. Необходимо привести описание последовательности разработки программы, реализации используемых методов, алгоритмов, блок-схем.

4. Анализ результатов. Привести анализ входных и выходных данных. Показать результаты выполнения программного кода. Предоставить скриншоты обработки тестовых примеров. Сделать выводы.

5. Листинг программы. Привести листинг разработанного программного кода, содержание файлов входных и выходных данных.

# Используемое программное обеспечение

1. Среда программирования MS Visual Studio Community 2022 (Свободно распространяемое программное обеспечение (в учебных целях));
2. Microsoft Office Standard 2007 (Open License: 42267924);
3. Open Office (Свободно распространяемое программное обеспечение).
4. Браузер (Свободно распространяемое программное обеспечение).

# Список литературы

* + - 1. Мейер Б. Объектно-ориентированное программирование и программная инженерия [Электронный ресурс]/ Мейер Б. – Электрон. текстовые данные. – М.: Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. – 285 c.
      2. Биллиг, В. A. Основы объектного программирования на С# (C# 3.0, Visual Studio 2008) [Электронный ресурс] : учебное пособие / В. A. Биллиг. — Электрон. текстовые данные. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 583 c. — 978-5-4487-0145-0. — Режим доступа: http://www.iprbookshop.ru/72339.html
      3. Павловская, Т. А. Программирование на языке высокого уровня C# [Электронный ресурс] / Т. А. Павловская. — 2-е изд. — Электрон. текстовые данные. — М. : Интернет-Университет Информационных Технологий (ИНТУИТ), 2016. — 245 c. — 2227-8397. — Режим доступа: http://www.iprbookshop.ru/73713.html
      4. Агапов, В. П. Основы программирования на языке С# [Электронный ресурс] : учебное пособие / В. П. Агапов. — Электрон. текстовые данные. — М. : Московский государственный строительный университет, ЭБС АСВ, 2012. — 128 c. — 978-5-7264-0576-6. — Режим доступа: http://www.iprbookshop.ru/16366.html
      5. Медведев, М. А. Программирование на СИ# [Электронный ресурс] : учебное пособие / М. А. Медведев, А. Н. Медведев ; под ред. А. В. Присяжный. — Электрон. текстовые данные. — Екатеринбург : Уральский федеральный университет, ЭБС АСВ, 2015. — 64 c. — 978-5-7996-1561-1. — Режим доступа: http://www.iprbookshop.ru/69667.html
      6. Казанский А.А. Объектно-ориентированное программирование на языке Microsoft Visual С# в среде разработки Microsoft Visual Studio 2008 и .NET Framework. 4.3 [Электронный ресурс]: учебное пособие и практикум/ Казанский А.А.— Электрон. текстовые данные.— М.: Московский государственный строительный университет, ЭБС АСВ, 2011.— 180 c
      7. Уйманова Н.А. Основы объектно-ориентированного программирования [Электронный ресурс]: практикум/ Уйманова Н.А., Таспаева М.Г.— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2017.— 156 c.— Режим доступа: http://www.iprbookshop.ru/78808.html.— ЭБС «IPRbooks»
      8. Новиков П.В. Объектно-ориентированное программирование [Электронный ресурс]: учебно-методическое пособие к лабораторным работам/ Новиков П.В.— Электрон. текстовые данные.— Саратов: Вузовское образование, 2017.— 124 c.— Режим доступа: http://www.iprbookshop.ru/64650.html.— ЭБС «IPRbooks»