# Практическое задание 1

Реализовать класс Person в котором будут

1. Свойства
   1. Фамилия (string)
   2. Имя (string)
   3. Отчество (string)
   4. Пол (bool)
   5. Рост/Вес (float)
   6. Дата рождения (DateTime)
   7. Возраст (int)

Все свойства должны быть public и read-only, свойства a,b,c,e,f автоматическими, свойство d реализовано через private поле, свойство g должно вычисляться при запросе.

1. Конструктор и методы Parse и ToString.
   1. public Person(string name, …)
   2. public static Person Parse(string text)
   3. public override string ToString()

В функции Main реализовать чтение данных из файла, создание массива персон, сортировку (на выбор пользователя - по убыванию или возрастанию) и вывод в данных в файл результатов в новом формате, в соответствии с вариантом.

1. Изменить реализацию используя тип record вместо класса. Сравнить удобство использования.

**Варианты**

|  | **Формат даты рождения** | **Вес / Рост** | **Поле сортировки** | **Формат числа (E или F)** | **Количество знаков после разделителя** |
| --- | --- | --- | --- | --- | --- |
| 1 | 01.09.2018 | Вес | Фамилия | E | 1 |
| 2 | 01.09.2018 | Вес | Дата рождения | E | 2 |
| 3 | 01.09.2018 | Вес | Фамилия | F | 3 |
| 4 | 09-01-2018 | Вес | Дата рождения | F | 4 |
| 5 | 09-01-2018 | Рост | Фамилия | E | 3 |
| 6 | 09-01-2018 | Рост | Дата рождения | E | 4 |
| 7 | 09/01/2018 | Рост | Фамилия | F | 3 |
| 8 | 09/01/2018 | Рост | Дата рождения | F | 4 |
| 9 | 09/01/2018 | Вес | Фамилия | E | 1 |
| 10 | 1 сен 2018 | Вес | Дата рождения | F | 2 |
| 11 | 1 сен 2018 | Вес | Фамилия | E | 3 |
| 12 | 1 сен 2018 | Вес | Дата рождения | F | 4 |
| 13 | 1 сентября 2018 | Рост | Фамилия | E | 3 |
| 14 | 1 сентября 2018 | Рост | Дата рождения | F | 4 |
| 15 | 1 сентября 2018 | Рост | Фамилия | E | 3 |
| 16 | 01-09-2018 | Рост | Дата рождения | F | 4 |
| 17 | 01-09-2018 | Вес | Фамилия | E | 1 |

# Практическое задание 2

1. Сделать интерфейс IPerson

interface IPerson

{

string Name { get; }

string Patronomic { get; }

string Lastname { get; }

DateTime Date { get; }

int Age { get; }

}

1. Сделать классы Student и Teacher реализующие этот интерфейс. Они должны содержать конструктор, принимающий значения свойств, статическую функцию создания из строки и переопределение функции ToString.
   1. У класса Student дополнительно должны быть свойства: курс, группа и средний балл.
   2. У класса Teacher дополнительно должны быть свойства: кафедра, стаж и должность, причем должность должна быть типом enum.
2. Сделать класс University, реализующий свойства и методы из следующего интерфейса.

interface IUniversity

{

IEnumerable<IPerson> Persons { get; } // отсортировать в соответствии с вариантом 1-й лабы

IEnumerable<Student> Students { get; } // отсортировать в соответствии с вариантом 1-й лабы

IEnumerable<Teacher> Teachers { get; } // отсортировать в соответствии с вариантом 1-й лабы

void Add(IPerson person);

void Remove(IPerson person);

IEnumerable<IPerson> FindByLastName(string lastName);

// Для нечетных вариантов. Выдать всех студентов, чей средний балл выше заданного.

// Отсортировать по среднему баллу

IEnumerable<Student > FindByAvrPoint(float avrPoint);

// Для четных вариантов. Выдать всех преподавателей, название кафедры которых содержит

// заданный текст. Отсортировать по должности.

IEnumerable<Teacher> FindByDepartment(string text);

}

1. Сделать головную программу с консольным меню, которая бы демонстрировала работу всего вышеперечисленного.

# Практическое задание 3

Сделать иерархию классов для вычисления математических выражений.

1. Базовый интерфейс

interface IExpr

{

double Compute(IReadOnlyDictionary<string, double> variableValues);

IEnumerable<string> Variables { get; }

bool IsConstant { get; }

bool IsPolynom { get; }

}

1. Абстрактные классы

UnaryOperation, BinaryOperation, Function

1. Классы реализующие арифметические операции и класс Variable, Constant.
2. Сделать для этих классов перегрузку операторов и ToString().
3. Классы реализующие функции (выбрать один, в соответствии с номером бригады)
   1. Степенные (в том числе показательные и логарифм)
   2. Тригонометрические
   3. Обратные тригонометрические
   4. Гиперболические
   5. Обратные гиперболические
4. Доп. задания
   1. Дифференцирование (3)
   2. Разбор выражений (4)
   3. Упрощение выражений (4)
   4. Интегрирование (4)
   5. Векторная арифметика (3)
   6. Добавление своих функций (2)

**Пример**

var a = new Variable("a");

var b = new Variable("b");

var expr0 = new Mult(new Add(a, b), new SinFunc(new Divide(a, new Constant(2))));

var expr = (a + b) \* Sin(a / 2);

Console.WriteLine(expr);

Console.WriteLine(expr.Compute(new Dictionary<string, double>{["a"] = 5, ["b"] = 3});