Самостоятельная работа от 12.02.2020

Задание 1

Создайте в Visual Studio проект консольного типа.

Определите в проекте ряд классов, перечисленных ниже в задании.

Требования:

* каждый класс определить в отдельном файле.
* каждый класс снабдить переопределенным методом ToString(), возвращающим строку с полным описанием данных экземпляра класса. Определите в каждом классе специальный конструктор, принимающий значения каждого поля/свойства класса
* в методе Main() создайте по одному экземпляру каждого определённого класса. Используя конструктор задайте каждому полю/свойству значения (адекватные, читаемые)
* используя метод ToString() получите строковое представление каждого объекта. Выведите в консоль полученное строковое представление каждого объекта.

Перечень классов:

1. Студент.   
   Свойства: Имя, Курс, Половая принадлежность (доступно только для чтения, т.е. readonly)
2. Служащий  
   Свойства: Имя, Профессия, Рабочий стаж
3. Цех  
   Свойства: Строковый шифр, количество служащих
4. Книга  
   Свойства: Название (Заголовок), Список фио авторов, Стоимость
5. Зачёт  
   Свойства: ФИО экзаменуемого, ФИО экзаменатора, Дата, Оценка
6. Адрес  
   Свойства: Почтовый индекс, Город, Улица, Дом, квартира
7. Товар  
   Свойства: Наименование, Количество, Стоимость, Срок годности
8. Учебная группа  
   Свойства: Шифр, Количество студентов, Год формирования, Специальность
9. Денежная купюра  
   Свойства: Серия и номер, Наминал, Наминал прописью

10. Компьютерная игра  
 Свойства: Название, Фирма разработчик, Год издания, Жанр

Листинг класса Program:

static void Main()

{

Student student = new Student("Иванов", 2, "Мужской");

Console.WriteLine(student.ToString());

Employee employee = new Employee("Иванов", "ИСП-1", 3);

Console.WriteLine(employee.ToString());

Shop shop = new Shop("ЛФ", 150);

Console.WriteLine(shop.ToString());

Book book = new Book("C#", new string[] { "Троелсан", "Котов" }, 5000);

Console.WriteLine(book.ToString());

Test test = new Test("Харитонов А.О.", "Иванова О.В.", "01.01.2020", 5);

Console.WriteLine(test.ToString());

Address address = new Address(443101, "Волжский", "Дружбы", "189", 13);

Console.WriteLine(address.ToString());

Product product = new Product("Книга", 15, 3000, "Отсутствует");

Console.WriteLine(product.ToString());

Group group = new Group("ИСП", 20, 2018, "ИСП-1");

Console.WriteLine(group.ToString());

Banknote banknote = new Banknote(303, 5, "Пять рублей");

Console.WriteLine(banknote.ToString());

Game game = new Game("CS:GO", "Valve", "2008", "Шутер");

Console.WriteLine(game.ToString());

Console.ReadKey();

}

Листинг класса Student:

public string Name { get; set; }

public int Course { get; set; }

public readonly string Gender;

public Student(string name, int course, string gender)

{

Name = name;

Course = course;

Gender = gender;

}

public override string ToString()

{

return $"Имя: {Name}; Курс: {Course}; Пол: {Gender}";

}

Листинг класса Employee:

public string Name { get; set; }

public string Profession { get; set; }

public int WorkExperience { get; set; }

public Employee(string name, string prof, int workexp)

{

Name = name;

Profession = prof;

WorkExperience = workexp;

}

public override string ToString()

{

return $"Имя: {Name}; Профессия: {Profession}; Стаж: {WorkExperience}";

}

Листинг класса Shop:

public string Cipher { get; set; }

public int Quantity { get; set; }

public Shop(string cipher, int quantity)

{

Cipher = cipher;

Quantity = quantity;

}

public override string ToString()

{

return $"Строковой шифр: {Cipher}; Количество служащий: {Quantity}";

}

Листинг класса Book:

public string Name { get; set; }

public string[] Authors { get; set; }

public int Price { get; set; }

public Book(string name, string[] authors, int price)

{

Name = name;

Authors = authors;

Price = price;

}

public override string ToString()

{

return $"Название: {Name}; Авторы: {(Authors != null ? string.Join(", ", Authors) : "")}; Цена: {Price}";

}

Листинг класса Test:

public string ExamineeFIO { get; set; }

public string ExaminerFIO { get; set; }

public string Date { get; set; }

public int Evalution { get; set; }

public Test(string examineefio, string examinerfio, string date, int evalution)

{

ExamineeFIO = examineefio;

ExaminerFIO = examinerfio;

Date = date;

Evalution = evalution;

}

public override string ToString()

{

return $"ФИО экзаменуемого: {ExamineeFIO}; ФИО экзаменатора: {ExaminerFIO}; Дата: {Date}; Оценка: {Evalution}";

}

Листинг класса Address:

public int PostalCode { get; set; }

public string City { get; set; }

public string Street { get; set; }

public string House { get; set; }

public int Flat { get; set; }

public Address(int postalcode, string city, string street, string house, int flat)

{

PostalCode = postalcode;

City = city;

Street = street;

House = house;

Flat = flat;

}

public override string ToString()

{

return $"Почтовый индекс: {PostalCode}; Город: {City}; Улица: {Street}; Дом: {House}; Квартира: {Flat}";

}

Листинг класса Product:

public string Name { get; set; }

public int Quantity { get; set; }

public double Price { get; set; }

public string ShelfLife { get; set; }

public Product(string name, int quantity, double price, string shelflife)

{

Name = name;

Quantity = quantity;

Price = price;

ShelfLife = shelflife;

}

public override string ToString()

{

return $"Имя: {Name}; Количество: {Quantity}; Цена: {Price}; Срок годности: {ShelfLife}";

}

Листинг класса Group:

public string Cipher { get; set; }

public int Quantity { get; set; }

public int YearOfFormation { get; set; }

public string Specialty { get; set; }

public Group(string cipher, int quntity, int yearformation, string specialty)

{

Cipher = cipher;

Quantity = quntity;

YearOfFormation = yearformation;

Specialty = specialty;

}

public override string ToString()

{

return $"Шифр: {Cipher}; Количество: {Quantity}; Год формирования: {YearOfFormation}; Специальность: {Specialty}";

}

Листинг класса Banknone:

public int Series { get; set; }

public int Nominal { get; set; }

public string NominalString { get; set; }

public Banknote(int series, int nominal, string nominalstring)

{

Series = series;

Nominal = nominal;

NominalString = nominalstring;

}

public override string ToString()

{

return $"Серия: {Series}; Номинал: {Nominal}; Номинал прописью: {NominalString}";

}

Листинг класса Game:

public string Name { get; set; }

public string Firm { get; set; }

public string PublicationDate { get; set; }

public string Genre { get; set; }

public Game(string name, string firm, string publdate, string genre)

{

Name = name;

Firm = firm;

PublicationDate = publdate;

Genre = genre;

}

public override string ToString()

{

return $"Имя: {Name}; Фирма: {Firm}; Год издания: {PublicationDate}; Жанр: {Genre}";

}

Результат работы программы:

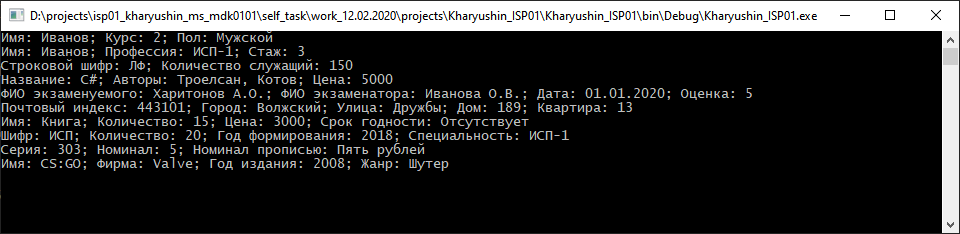


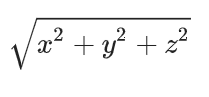
Рисунок 1 - Результат работы программы задания 1

Задание 2

Создайте в Visual Studio проект консольного типа.

Определите в проекте класс, описывающий вектор в трёхмерном пространстве.

Класс должен включать:

* конструктор с параметрами в виде списка координат x, y, z
* метод, вычисляющий длину вектора по формуле:   
  
* метод, вычисляющий скалярное произведение этого вектора с другим по формуле:  
  
* метод, вычисляющий векторное произведение с другим вектором:  
  
* Переопределение метода ToString(), возвращающий строковое представление вектора в формате: vec (x:число ; y:число ; z:число)

В методе Main() создайте два экземпляра класса, описывающего вектор. Используя конструктор задайте для каждого экземпляра, его каждому полю/свойству значения. Используя метод ToString() получите строковое представление каждого объекта. Выведите в консоль полученные строковые представления объектов.  
Рассчитайте модули (длины) векторов и выведите в консоль результат с текстовым сообщением, поясняющим, что это длины векторов.  
Рассчитайте скалярное произведение векторов и выведите в консоль результат с текстовым сообщением, поясняющим, что это результат скалярного произведения векторов.

Рассчитайте векторное произведение векторов и выведите в консоль результат с текстовым сообщением, поясняющим, что это результат векторного произведения векторов.

Листинг класса Program:

static void Main()

{

Vector3D vector01 = new Vector3D(5, 10, 3);

Console.WriteLine(vector01.ToString());

Vector3D vector02 = new Vector3D(15, 2, 25);

Console.WriteLine(vector02.ToString());

Console.WriteLine();

Console.WriteLine($"Длина 1-го вектора: {vector01.Length()}; Длина 2-го вектора: {vector02.Length()}");

Console.WriteLine($"Скалярное произведение векторов: {vector01.ScalarProductVectors(vector02)}");

Console.WriteLine($"Векторное произведение векторов: {vector01.VectorsProduct(vector02)}");

Console.ReadKey();

}

Листинг класса Vector3D:

public double X { get; set; }

public double Y { get; set; }

public double Z { get; set; }

public Vector3D(double x, double y, double z)

{

X = x;

Y = y;

Z = z;

}

public double Length()

{

return Math.Sqrt((Math.Pow(this.X, 2)) + Math.Pow(this.Y, 2) + Math.Pow(this.Z, 2));

}

public double ScalarProductVectors(Vector3D other)

{

return (this.X \* other.X) + (this.Y \* other.Y) + (this.Z \* other.Z);

}

public Vector3D VectorsProduct(Vector3D other)

{

return new Vector3D ( (this.Y \* other.Z) - (this.Z \* other.Y),

(this.Z \* other.X) - (this.X \* other.Z),

(this.X \* other.Y) - (this.Y \* other.X) );

}

public override string ToString()

{

return $"vec (x:{X} ; y:{Y} ; z:{Z})";

}

Результат работы программы:

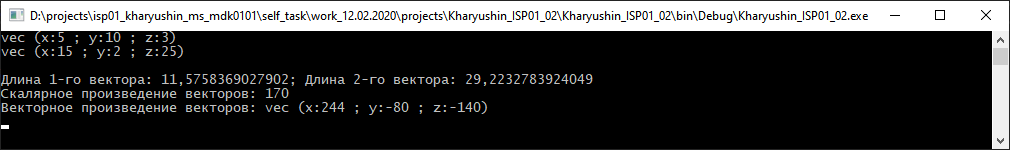


Рисунок 2 - Результат работы программы задания 2