Частное учреждение образования

«Колледж бизнеса и права»

ОТЧЕТ ПО УЧЕБНОЙ ПРАКТИКЕ ПО ПРОГРАММИРОВАНИЮ

ОП Т.792008

Руководитель практики Е.В. Багласова

Учащейся Кремко О.Н.

2020

Содержание

Изм.

Лист

№ докум.

Подпись

Дата

Лист

2

ОП Т.792008

Разраб.

*Кремко О.Н.*

Провер.

Багласова Е.В.

Т. контр.

Н. контр.

Утверд.

Отчет по практике по программированию

Лит.

Листов

КБП

[Введение 3](#_Toc26887818)

[1 Объектно-ориентированное программирование с использованием языка C# 4](#_Toc26887819)

[1.1 Организация циклов 4](#_Toc26887820)

[1.2 Массивы 5](#_Toc26887821)

[1.3 Базовые операции над строками 6](#_Toc26887822)

[1.4 Регулярные выражения 8](#_Toc26887823)

[1.5 Структуры 9](#_Toc26887824)

[1.6 Простейшие классы 10](#_Toc26887825)

[1.7 Классы и операции 11](#_Toc26887826)

[1.8 Наследование 15](#_Toc26887827)

[1.9 Делегаты, события, лямбда-выражения 17](#_Toc26887828)

[1.10 Коллекции 19](#_Toc26887829)

[1.11 Интерфейсы 20](#_Toc26887830)

[1.12 Обобщения 22](#_Toc26887831)

[1.13 LINQ 22](#_Toc26887832)

[1.14 Работа с объектами файловой системы 22](#_Toc26887833)

[2 Разработка приложений средствами библиотеки классов Microsoft .NET Framework 24](#_Toc26887834)

[3 Программирование быстродействующих информационных систем 25](#_Toc26887835)

[Заключение 26](#_Toc26887836)

[Список информационных источников 27](#_Toc26887837)

# Введение

Целью учебной практики по программированию является формирование и закрепление навыков разработки объектно-ориентированных программ.

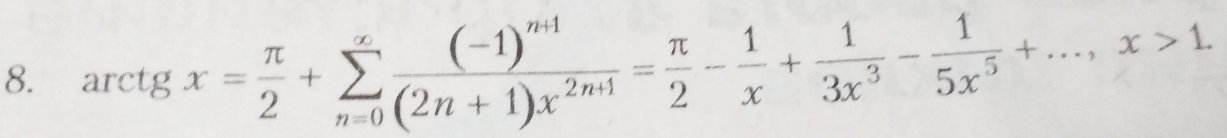
Задачами практики являются:

* усиление взаимосвязи теоретического обучения с будущей профессиональной деятельностью;
* углубление знаний учащихся по дисциплине «Конструирование программ и языки программирования»;
* формирование первичных умений и навыков по специальности.

# Объектно-ориентированное программирование с использованием языка C#

## Организация циклов

Вычислить и вывести на экран в виде таблицы значения функции, заданной с помощью ряда Тейлора, на интервале от *xнач* до *xкон* с шагом *dx* с точностью *e*. Таблицу снабдить заголовком и шапкой. Каждая строка таблицы должна содержать значение аргумента, значение функции и количество просуммированных членов ряда.



class Program

{

static void Main(string[] args)

{

double x, eps, Xstart, Xend, dx;

Console.WriteLine("Vvedite X nahalinoe");

string buf\_Xstart = Console.ReadLine();

Console.WriteLine("Vvedite Xkonechnoe");

string buf\_Xend = Console.ReadLine();

Console.WriteLine("Vvedite E");

string buf\_Eps = Console.ReadLine();

Console.WriteLine("Vvedite shag");

string buf\_dx = Console.ReadLine();

eps = Convert.ToDouble(buf\_Eps);

Xstart = Convert.ToDouble(buf\_Xstart);

Xend = Convert.ToDouble(buf\_Xend);

dx = Convert.ToDouble(buf\_dx);

Console.WriteLine("-----------------------------------");

Console.WriteLine(" x | F | n | Actg(x) ");

for (x = Xstart; x <= Xend; x += dx)

{

int n=0;

double r=1, y = Math.PI/2;

double c = (1 - x);

while(Math.Abs(r) > eps)

{

r = Math.Pow(-1, n + 1) / ((2\*n+1)\*Math.Pow(x, 2\*n+1));

y += r;

n++;

}

Console.WriteLine($"{x,6:f1} | {y,20:f15} | {n,4} | {Math.Atan(x),20:f15}" );

}

Console.ReadKey();

}

}

Результат программы представлен на рисунке 1

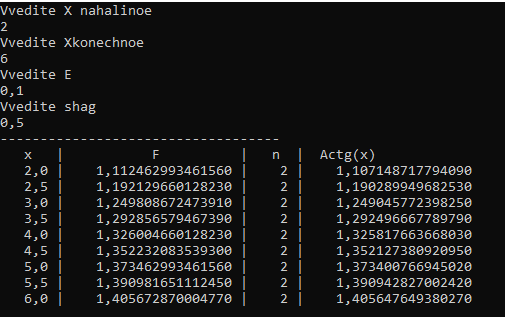


Рисунок 1

## Массивы

Дано натуральное число n, целые числа А1,..., А10 и B1,..., Bn. Среди А1,..., А10 нет чисел, которые повторяются, отсутствуют они и среди B1,..., Bn. Построить объединение данных последовательностей, в котором нет одинаковых чисел.

int[] a = new int[] { 10, 11, 12, 15, 17, 25, 23, 33 };

int[] b = new int[] { 10, 13, 15, 17, 25, 29, 27, 24};

int i = 0;

Console.WriteLine("Массив А: ");

for (i = 0; i < a.Length; i++)

{

Console.Write(" " + a[i]);

}

Console.WriteLine();

Console.WriteLine("Массив Б: ");

for (i = 0; i < b.Length; i++)

{

Console.Write(" " + b[i]);

}

Console.WriteLine();

Console.WriteLine("Не повторяющиеся элементы: ");

int[] c = new int[a.Length+b.Length];

int k = 0;//index of c: c[k]

int count = 0;//repeat's count

for (i = 0; i < a.Length; i++)

{

count = 0;

for (int j=0;j<b.Length;j++)

if (a[i] == b[j])

{

count++;

}

if (count == 0) { c[k] = a[i]; k++; }

}

int[] c1 = new int[a.Length + b.Length];

for (i = 0; i < b.Length; i++)

{

count = 0;

for (int j = 0; j < a.Length; j++)

if (b[i] == a[j])

{

count++;

}

if (count == 0) { c[k] = b[i]; k++; }

}

Console.WriteLine();

for(i=0;i<k;i++)

Console.Write(" " + c[i]);

Console.WriteLine();

Console.ReadKey();

}

}

Результат работы программы представлен на рисунке 2

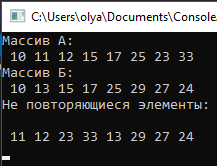


Рисунок 2

## Базовые операции над строками

Дан текст, в котором слова разделяются пробелом, а в конце стоит восклицательный знак (!). необходимо:

а) подсчитать количество слов в тексте i вывести на экран те слова, длина которых меньше данный счет d;

б) определить, какой процент слов в тексте имеет самую короткую длину;

в) исключить из текста слова, которые имеют минимальную длину, остальное текст

сдвинуть.

class Program

{

static void Main(string[] args)

{

string text = " знаю, что придумать. Пусть это будет текст!";

char[] a = text.ToCharArray();

char[] b = { };

Console.WriteLine("Текст: ");

for (int i = 0; i < a.Length; i++)

{

Console.Write(a[i]);

}

Console.WriteLine();

Console.WriteLine("Всего слов: ");

int count = 0;

for (int i = 0; i < a.Length; i++)

{

if (a[i] == ' ')

{ count++; }

}

Console.WriteLine(count);

Console.WriteLine("Ввведите число букв: ");

string buf = Console.ReadLine();

int d = Convert.ToInt32(buf);

var str = " знаю, что придумать. Пусть это будет текст!";

char[] sep = new char[] { ' ', ',' };

string[] s = str.Split(sep, StringSplitOptions.RemoveEmptyEntries);

string min\_word = str;

Console.WriteLine("Слова, число букв которых меньше заданного числа букв: ");

foreach (var x in s)

{

if (x.Length < d)

{

Console.Write(x + " ");

}

}

Console.WriteLine();

Console.WriteLine("Без которких слов: ");

int min = 999999999;

foreach (var x in s)

{

if (min\_word.Length > x.Length)

{

min\_word = x;

min = min\_word.Length;

}

}

int temp = 0;

foreach (var x in s)

{

if (min < x.Length)

{

Console.Write(x + " ");

temp++;

}

}

Console.WriteLine();

double procent = 100 - ((temp \* 100) / count);

Console.WriteLine("Короткие слова занимают " + procent + "%");

Console.ReadKey();

}

}

Результат работы программы представлен на рисунке 3

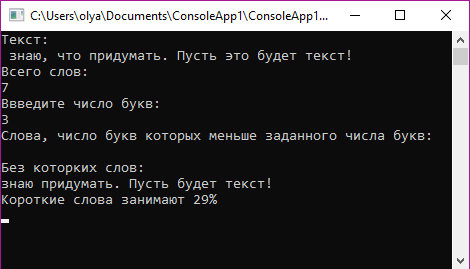


Рисунок 3

## Регулярные выражения

Найти количество строк текстового файла, которые не содержат слов, являющихся правильной записью времени в 12-часовом формате (часы и минуты разделяются двоеточием, секунды отсутствуют).

class Program

{

static void Main()

{

using (TextReader text = File.OpenText("стих.txt"))

{

while (text.Peek() > -1)

{

string[] lines = File.ReadAllLines("стих.txt");

string pattern = @"(1([0-2]|[1-9])):[0-5][0-9]\s(am|pm)";

int count = 0;

foreach (string value in lines)

{

if (Regex.IsMatch(value, pattern))

{

Console.WriteLine("По формату: " + value);

}

else

{

Console.WriteLine("Не по формату: " + value);

count++;

}

}

Console.WriteLine("количество строк, которые не содержат правильных слов в 12-часовом формате: " + count);

Console.ReadKey();

}

}

Console.ReadKey();

}

}

Результат работы программы представлен на рисунке 4

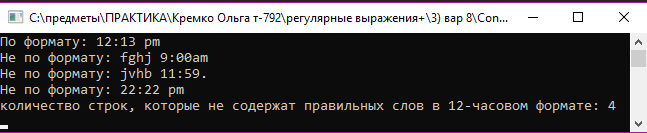


Рисунок 4

## Структуры

Информация об участниках спортивных соревнований содержит наименование страны, название команды, ФИО игрока, игровой номер, возраст, рост, вес. Вывести информацию о самом молодом участнике.

struct SportGamer

{

public string CountryName;

public string TeamName;

public string PlayerName;

public int PlayerNum;

public int PlayerAge;

public double PlayerHeight;

public double PlayerWeight;

public SportGamer(string CountryName, string TeamName, string PlayerName, int PlayerNum, int PlayerAge, double PlayerHeight, double PlayerWeight)

{

this.CountryName = CountryName;

this.TeamName = TeamName;

this.PlayerName = PlayerName;

this.PlayerNum = PlayerNum;

this.PlayerAge = PlayerAge;

this.PlayerHeight = PlayerHeight;

this.PlayerWeight = PlayerWeight;

}

public void inf ()

{

Console.WriteLine(" Наименование страны {0};\n Название команды{1};\n" +

" ФИО игрока {2};\n Игровой номер {3};\n" + " Возраст {4};\n" +

" Рост {5};\n Вес {6};\n ", CountryName, TeamName, PlayerName,

PlayerNum, PlayerAge, PlayerHeight, PlayerWeight);

}

};

class Class1

{

static void Main()

{

SportGamer a = new SportGamer("Россия", " КомандаРоссия", "Фамилия И.И.", 1, 30, 1.87, 75);

SportGamer b = new SportGamer("Литва", " КомандаЛитва", " Фамили.Я.", 2, 29, 1.90, 74);

SportGamer c = new SportGamer("Украина", "КомандаУкраина", "Фамил М.С.", 3, 31, 1.78, 73);

SportGamer d = new SportGamer("Беларусь", "КомандаБеларусь", "Фами В.Г.", 4, 30, 1.91, 77);

List<SportGamer> all = new List<SportGamer> { a, b, c, d };

var sort = from u1 in all

orderby u1.PlayerAge

select u1;

foreach (SportGamer u1 in sort)

{

u1.inf();

break;

}

Console.ReadKey();

Console.ReadKey();

}

}

Результат работы программы представлен на рисунке 5

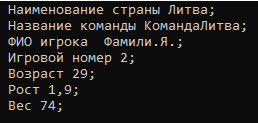


Рисунок 5

## Простейшие классы

Каждый разрабатываемый класс должен, как правило, содержать следующие элементы: скрытые поля, конструкторы с параметрами и без параметров, методы, свойства. Свойства и методы должны обеспечивать непротиворечивый, полный, минимальный и удобный интерфейс класса. При возникновении ошибок должны выбрасываться исключения.

Cоздать класс с указанными двумя полями (вещественное число – первый катет прямоугольного треугольника, вещественное число – второй катет прямоугольного треугольника)и тремя методами:

* конструктор для инициализации объекта;
* функция формирования строки с информацией об объекте;
* функция вычисления площади прямоугольного треугольника.

Продемонстрировать работоспособность разработанного класса.

class Tringle

{

public double Katet\_1;

public double Katet\_2;

public Tringle()

{

Katet\_1 = 0;

Katet\_2 = 0;

}

public Tringle(double k1, double k2)

{

Katet\_1 = k1;

Katet\_2 = k2;

}

public double Tringle\_square(double a, double b)

{

return (a \* b) / 2;

}

public string ToShortString()

{

return "Катет 1: " + Katet\_1 + "\n" + "Катет 2: " + Katet\_2 + "\n" + "Площадь треугольника: " + Tringle\_square(Katet\_1, Katet\_2);

}

static void Main(string[] args)

{

Tringle a1 = new Tringle();

a1.Katet\_1 = 7;

Tringle a2 = new Tringle();

a1.Katet\_2 = 8;

Console.WriteLine(a1.ToShortString());

Console.ReadKey();

}

Результат работы программы представлен на рисунке 6

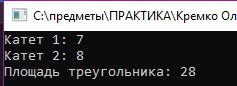


Рисунок 6

## Классы и операции

Каждый разрабатываемый класс должен, как правило, содержать следующие элементы: скрытые поля, конструкторы с параметрами и без параметров, методы, свойства, индексаторы, перегруженные операции. Функциональные элементы класса должны обеспечивать непротиворечивый, полный, минимальный и удобный интерфейс класса. При возникновении ошибок должны выбрасываться исключения. Продемонстрировать использование всех разработанных элементов классов.

Описать пользовательский класс: ядро атома, протон, нейтрон, электорон, химический элемент, ион.

class ChemicaElelement

{

public string ion;

public double AtomCore;

public double Proton;

public double Neytron;

public double Electron;

public int num\_in\_table;

public double AtomM;

public int valentnost;

public ChemicaElelement ()

{

this.ion = "NoName";

this.AtomCore = 0;

this.Electron = 0;

this.Neytron = 0;

this.Proton = 0;

this.AtomM = 0;

}

public ChemicaElelement (double AtomCore, double Proton, double Neytron, double Electron)

{

this.AtomCore = AtomCore;

this.Electron = Electron;

this.Neytron = Neytron;

this.Proton = Proton;

//this.valentnost = valentnost;

}

public string AnName

{

get { return ion; }

set { ion = value; }

}

public double A\_AtomCore

{

get { return AtomCore; }

set { AtomCore = value; }

}

public double A\_Electron

{

get { return Electron; }

set { Electron = value; }

}

public double A\_Neytron

{

get { return Neytron; }

set { Neytron = value; }

}

public double A\_Proton

{

get { return Proton; }

set { Proton = value; }

}

public double Result1 (int num\_in\_table)

{

double AtomCore = num\_in\_table;

return AtomCore;

}

public double Result2 (int num\_in\_table)

{

double Electon = 0 - num\_in\_table;

return Electon;

}

public double Result3(int num\_in\_table, double AtomM)

{

double Neytron = AtomM - num\_in\_table;

return Neytron;

}

public double Result3()

{

double Neytron = AtomM - num\_in\_table;

return Neytron;

}

public double Result4(int num\_in\_table)

{

double Proton = num\_in\_table;

return Proton;

}

public void result ()

{

Console.WriteLine("Введите условное обозначение элемента: ");

ion = Console.ReadLine();

Console.WriteLine("Введите номер элемента из таблицы Менделеева: ");

string buf2 = Console.ReadLine();

num\_in\_table = Convert.ToInt32(buf2);

Console.WriteLine("Введите атомную массу элемента из таблицы Менделеева: ");

string buf3 = Console.ReadLine();

AtomM = Convert.ToDouble(buf3);

if (num\_in\_table > 120)

{

Console.WriteLine("Элемента с таким номером не существует.");

}

else

{

Console.WriteLine("Заряд атома: +" + Result1(num\_in\_table));

Console.WriteLine("Электронов: " + Result2(num\_in\_table));

Console.WriteLine("Нейтронов: " + Result3(num\_in\_table, AtomM));

Console.WriteLine("Протонов: +" + Result4(num\_in\_table));

}

}

}

class ChemicalReaction : ChemicaElelement

{

public ChemicalReaction(string name)

{

this.ion = "CI";

this.valentnost = 1;

}

public string Name

{

get { return ion; }

set { ion = value; }

}

}

class Res

{

static void Main()

{

Console.WriteLine("\*1 элемет\*");

ChemicaElelement a = new ChemicaElelement();

a.result();

Console.WriteLine("Взаимодействует ли этот ион с хлором?");

if (a.ion != "Ag" && a.ion != "Hg" )

{

Console.WriteLine("Вещество растворимо");

}

else

{

Console.WriteLine("Вещество не растворимо");

}

Console.ReadKey();

}

}

Результат работы программы предоставлен на рисунке 7

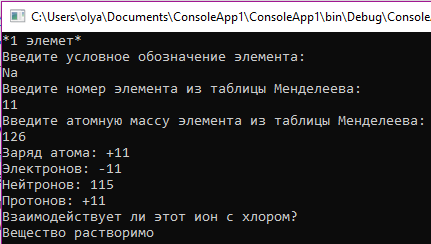


Рисунок 7

## Наследование

Построить три класса (базовый и 2 потомка), описывающих некоторых работников с почасовой оплатой (один из потомков) и фиксированной оплатой (второй потомок). Описать в базовом классе абстрактный метод для расчета среднемесячной заработной платы. Для «повременщиков» формула для расчета такова: «среднемесячная заработная плата = 20.8 \* 8 \* почасовую ставку», для работников с фиксированной оплатой «среднемесячная заработная плата = фиксированной месячной оплате». Упорядочить всю последовательность работников по убыванию среднемесячного заработка. При совпадении зарплаты – упорядочивать данные по алфавиту по имени. Вывести идентификатор работника, имя и среднемесячный заработок для всех элементов списка.

abstract class ZP

{

public double ZPPovremennaya;

public double ZPFix;

public double PochasovayaStavka;

public double FixPlata;

public virtual double ALLZP ()

{

return 0;

}

public virtual void inf() {}

public virtual void inf\_num() { }

}

class FIx : ZP

{

public int Wnumber;

public string WName;

public FIx()

{

Wnumber = 0;

WName = "Нет иформации о имени";

ZPFix = 0;

}

public FIx(int WNum, string WN, double Fpl)

{

WName = WN;

Wnumber = WNum;

FixPlata = Fpl;

}

public override double ALLZP()

{

ZPFix = FixPlata;

return ZPFix;

}

public override void inf()

{

Console.WriteLine(" Номер: {0}\n Имя: {1}\n Тип: фиксированная оплата труда\n Зарплата: {2}", Wnumber, WName, ALLZP() + "\n");

}

public override void inf\_num()

{

Console.WriteLine(" Номер: " + Wnumber);

}

}

class Povremen : ZP

{

public int Wnumber;

public string WName;

public Povremen()

{

Wnumber = 0;

WName = "Нет иформации о имени";

PochasovayaStavka = 0;

}

public Povremen(int WNum, string WN, double Pst)

{

Wnumber = WNum;

WName = WN;

PochasovayaStavka = Pst;

}

public override double ALLZP()

{

ZPPovremennaya = 20.8 \* 8 \* PochasovayaStavka;

return ZPPovremennaya;

}

public override void inf ()

{

Console.WriteLine(" Номер: {0}\n Имя: {1}\n Тип: повременная оплата труда\n Зарплата: {2}", Wnumber, WName, ALLZP() + "\n");

}

public override void inf\_num()

{

Console.WriteLine(" Номер: " + Wnumber);

}

}

class Program

{

static void Main(string[] args)

{

ZP a1 = new Povremen(1, "Работник20", 80.5);

ZP a2 = new Povremen(2, "Работник11", 70.3);

ZP a3 = new Povremen(3, "Работник15", 75.2);

ZP a4 = new Povremen(4, "Работник1", 81.2);

ZP b1 = new FIx(5, "Работник 5", 1060.4);

ZP b2 = new FIx(6, "Работник 6", 1063.4);

ZP b3 = new FIx(7, "Работник 7", 1060.4);

ZP b4 = new FIx(8, "Работник 8", 1072.4);

List<ZP> obj = new List<ZP>() { a1, a2, a3, a4, b1, b2, b3, b4 };

var sort1 = from u1 in obj

orderby u1.ALLZP()

select u1;

Console.WriteLine("Сортировка по ЗП: ");

foreach (ZP u1 in sort1)

{

u1.inf();

}

Console.ReadKey();

}

}

Результат работы программы представлен на рисунке 8

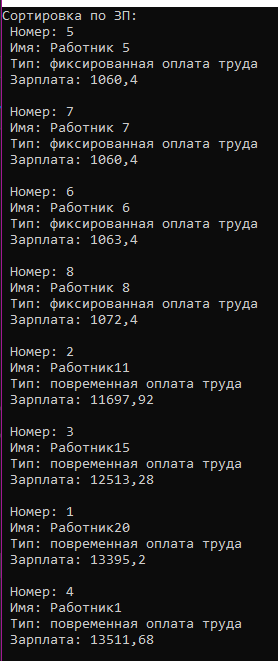


Рисунок 8

## Делегаты, события, лямбда-выражения

Реализовать метод, вычисляющий значение функции с заданным диапазоном (интервалом для вычислений) и заданным шагом. Виды функций, передаваемых в качестве параметра оформить в виде делегата. Интервал и шаг задаются произвольным образом. Функции: Atan2(x) и cosh(x).

class Program

{

public static void tabl1(funk F, double x, double y, double b)

{

Console.WriteLine("|X\_\_\_\_\_\_|Y\_\_\_\_\_\_|\_F(x,y)\_");

while (x <= b)

{

Console.WriteLine("| " + x + "\t|" + y + "\t|" + Math.Round(F(x,y), 3));

x += 1; //шаг

}

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

public static void tabl2(funk2 F, double x, double b)

{

Console.WriteLine("|X\_\_\_\_\_\_|\_\_\_\_F(x)\_\_\_\_\_\_\_\_");

while (x <= b)

{

Console.WriteLine("| " + x + "\t|" + Math.Round(F(x),3));

x += 1; // шаг

}

Console.WriteLine("\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_");

}

static void Main()

{

double b = 5;

double x = 0;

double y = 0.3;

Console.WriteLine(" Предел значения x: " + b);

Console.WriteLine(" x = " + x);

Console.WriteLine(" у = " + y);

Console.WriteLine(" Таблица функции Atan2(x) : ");

tabl1(new funk(Math.Atan2), x, y, b);

Console.WriteLine();

Console.WriteLine(" Таблица функции Cosh(x): ");

tabl2(new funk2(Math.Cosh), x, b);

Console.ReadKey();

}

}

Результат работы программы представлен на рисунке 9

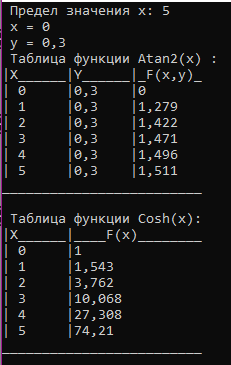


Рисунок 9

## Коллекции

В текстовом файле записана без ошибок формула вида: <формула>=<цифра>|M(<формула>,<формула>)|m(<формула>,<формула>) <цифра>=0|1|2|3|4|5|6|7|8|9. M обозначает вычисление максимума, m – минимума.

Решить с помощью Staсk.

class Program

{

static int min(int a, int b)

{

if (a < b)

{

return a;

}

else

{

return b;

}

}

static int max(int a, int b)

{

if (a < b)

{

return b;

}

else

{

return a;

}

}

static void Main()

{

Stack<int> numbers = new Stack<int>();

using (TextReader text = File.OpenText("текст.txt"))

{

char[] line = File.ReadAllText("текст.txt").ToCharArray();

for (int i = 0; i < line.Length; i++)

{

if (line[i] == '0'|| line[i] == '1' || line[i] == '2' || line[i] == '3'

|| line[i] == '4' || line[i] == '5' || line[i] == '6' ||

line[i] == '7' || line[i] == '8' || line[i] == '9')

{

numbers.Push(line[i]);

Console.Write(line[i] + " ");

}

}

}

Console.WriteLine();

int min\_ = min((char)numbers.Pop(), (char)numbers.Pop());

int max\_ = min((char)numbers.Pop(), (char)numbers.Pop());

int res = max(min\_, max\_);

Console.WriteLine("Результат M(m(3, 5),M(1, 2)) = " + (char)res);

Console.ReadKey();

}

Результат работы программы представлен на рисунке 10



Рисунок 10

## Интерфейсы

Реализовать иерархию классов из индивидуального задания. Создать интерфейс(-ы) для вычисления площади (если это возможно) и объема (если это возможно) фигуры. В интерфейсе также нужно определить свойство(-а), связанные с соответствующими данными классов.

Фигура - Квадрат – Треугольник (данные – стороны).

interface Intrf

{

double square();

void inf();

}

lass SQ : Intrf

{

public double a;

public SQ()

{

a = 0;

}

public SQ(double a\_)

{

a = a\_;

}

public double square()

{

return a \* a;

}

void Intrf.inf() // 2 сп

{

Console.WriteLine(" Сторона квадрата: a = {0};\n Площадь квадрата = {1}\n ", a, square());

}

}

class Tringle : Intrf

{

public double a;

public double b;

public double c;

public Tringle()

{

a = 0;

b = 0;

c = 0;

}

public Tringle(double a\_, double b\_, double c\_)

{

a = a\_;

b = b\_;

c = c\_;

}

public double square()

{

double p = (a + b + c) / 2;

double square = Math.Sqrt(p \* (p - a) \* (p - b) \* (p - c));

return square;

}

public void inf()

{

Console.WriteLine(" Стороны треугольника: a = {0}; b = {1}; c = {2};\n Площадь треугольника = {3}\n ", a, b, c, square());

}

}

class Program

{

static void Main(string[] args)

{

Intrf abcd = new SQ(3); // 2 сп

abcd.square();

abcd.inf();

Tringle abc = new Tringle(3, 4, 5);

abc.square();

abc.inf();

Console.ReadKey();

}

}

Результат работы программы представлен на рисунке 11

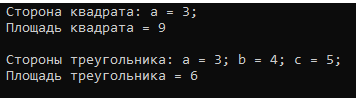


Рисунок 11

## Обобщения

Разработать обобщенный класс, позволяющий работать с данными на примере стека/очереди. Продемонстрировать его работу.

class Program

{

static void Main(string[] args)

{

Person<string> person1 = new Person<string> {Name = "Timmy", Surname = "Blanck" };

person1.Age = "34"; string age1 = person1.Age;

person1.Weight = "69,5"; string weight1 = person1.Weight;

person1.Height = "177"; string height1 = person1.Height;

Person<double> person2 = new Person<double> { Name = "Tonny", Surname = "Wonk" };

person2.Age = 27; double age2 = person2.Age;

person2.Weight = 83.12; double weight2 = person2.Weight;

person2.Height = 183.2; double height2 = person2.Height;

Person<int> person3 = new Person<int> { Name = "Vano", Surname = "Ilano" };

person3.Age = 14; int age3 = person3.Age;

person3.Weight = 55; int weight3 = person3.Weight;

person3.Height = 160; int height3 = person3.Height;

Person<float> person4 = new Person<float> { Name = "Ohara", Surname = "Tohara" };

person4.Age = 23; float age4 = person4.Age;

person4.Weight = 46.12f; float weight4 = person4.Weight;

person4.Height = 157.25f; float height4 = person4.Height;

Stack<string> st1 = new Stack<string>();

person1.Add(person1.Age, person1.Weight, person1.Height, st1);

Console.WriteLine($"{person1.Name} {person1.Surname}:");

foreach(string p in st1)

{

Console.WriteLine($"{p} ");

}

Console.WriteLine();

Stack<double> st2 = new Stack<double>();

person2.Add(person2.Age, person2.Weight, person2.Height, st2);

Console.WriteLine($"{person2.Name} {person2.Surname}:");

foreach (double p in st2)

{

Console.WriteLine($"{p} ");

}

Console.WriteLine();

Stack<int> st3 = new Stack<int>();

person3.Add(person3.Age, person3.Weight, person3.Height, st3);

Console.WriteLine($"{person3.Name} {person3.Surname}:");

foreach (int p in st3)

{

Console.WriteLine($"{p} ");

}

Console.WriteLine();

Stack<float> st4 = new Stack<float>();

person4.Add(person4.Age, person4.Weight, person4.Height, st4);

Console.WriteLine($"{person4.Name} {person4.Surname}:");

foreach (float p in st4)

{

Console.WriteLine($"{p} ");

}

Console.ReadKey();

}

}

class Person <T>

{

public string Name { get; set; }

public string Surname { get; set; }

public T Age { get; set; }

public T Weight { get; set; }

public T Height { get; set; }

public void Add(T Age, T Weight, T Height, Stack<T> stack)

{

this.Age = Age; this.Weight = Weight; this.Height = Height;

stack.Push(Age); stack.Push(Weight); stack.Push(Height);

}

}

Результат работы программы представлен на рисунке 12



Рисунок 12

## LINQ

В одномерном массиве, состоящем из п вещественных элементов, вычислить: номер минимального элемента массива; сумму элементов массива, расположенных между первым и вторым отрица­тельными элементами. Преобразовать массив таким образом, чтобы сначала располагались все элементы, модуль которых не превышает единицу, а потом — все остальные. Действия выполнить с помощью LINQ-запросов.

class Class1

{

static void Main()

{

int[] a = { 1, 12, -1, 10, 0, 120, 1, 8, 34, 56 };

Console.WriteLine("Исходный массив: ");

for (int i = 0; i < a.Length; i++)

{

Console.Write(a[i] + "\t");

}

Console.WriteLine();

Console.WriteLine("Минимальный: " + a.Min() + " индекс: " + Array.IndexOf(a, a.Min()));

var x = a.SkipWhile(i => i >= 0).Skip(1).TakeWhile(j => j >= 0).Sum();

Console.WriteLine("сумма элементов массива между 1ым и 2ым отрицательными элементами:" + x);

Console.WriteLine("Сортировка: ");

var result = a.Where(i => i >= -1 & i <= 1).Concat(a.Where(i => i < -1 || i > 1));

foreach (var item in result)

{

Console.Write($"{item,4}");

}

Console.ReadKey();

}

}

Результат программы представлен на рисунке 13

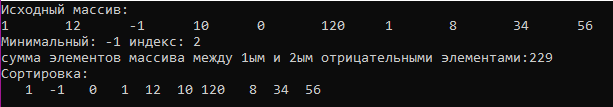


Рисунок 13

## Работа с объектами файловой системы

Дан текстовый файл. Напечатать первый символ каждой строки.

class Program

{

static void Main()

{

using (TextReader text = File.OpenText("стих.txt"))

{

while (text.Peek() > -1)

{

Console.WriteLine(text.ReadLine()[0]);

}

}

Console.ReadKey();

}

}

Результат работы программы представлен на рисунке 14

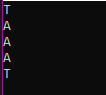


Рисунок 14

# Разработка приложений средствами библиотеки классов Microsoft .NET Framework

class Article : IRateAndCopy

{

public Person AuthorArticteInf { get; set; }

public string ArticleName { get; set; }

public double reting { get; set; }

public double Rating => throw new NotImplementedException();

double IRateAndCopy.Rating => throw new NotImplementedException();

public Article(Person AAI, string AN, double reting)

{

this.AuthorArticteInf = AAI;

this.ArticleName = AN;

this.reting = reting;

}

public Article()

{

this.AuthorArticteInf = new Person();

this.ArticleName = "Нет информации";

this.reting = 0;

}

public virtual object DeepCopy()

{

Article klon = new Article(AuthorArticteInf, ArticleName, reting);

return klon;

}

public override string ToString()

{

return string.Format(" Данные автора статьи:\n{0} Название статьи: {1},\n Рейтинг: {2}.\n", AuthorArticteInf, ArticleName, reting);

}

object IRateAndCopy.DeepCopy()

{

Article klon = new Article(AuthorArticteInf, ArticleName+"IDC", reting);

return klon;

}

}

lass Edition

{

protected string EditionName;

protected DateTime ReleseDate;

protected int PrintEdition;

ArrayList MagazineEditorList = new ArrayList();

ArrayList Magazine= new ArrayList();

public Edition (string EditionName, DateTime ReleseDate, int PrintEdition)

{

this.EditionName = EditionName;

this.ReleseDate = ReleseDate;

this.PrintEdition\_ = PrintEdition;

}

public Edition ()

{

this.EditionName = "Нет информации о названии издания";

this.ReleseDate = new DateTime(2019,01,01);

this.PrintEdition = 0;

}

public void FMagazineEditorsList(params Article[] a)

{

MagazineEditorList.AddRange(a);

}

public string EditionName\_

{

get { return EditionName; }

set { EditionName = value; }

}

public DateTime ReleseDate\_

{

get { return ReleseDate; }

set { ReleseDate = value; }

}

public virtual object DeepCopy()

{

Edition klon = new Edition(EditionName, ReleseDate, PrintEdition);

return klon;

}

public int PrintEdition\_

{

get { return PrintEdition; }

set

{

if (value < 0)

throw new ArgumentException("Присваемое значение отрицательно!!!");

else

PrintEdition = value;

}

}

public virtual bool Equals(object obj)

{

Edition edition = (Edition)obj;

if (EditionName == edition.EditionName && ReleseDate == edition.ReleseDate && PrintEdition == edition.PrintEdition)

return true;

else

return false;

}

public override int GetHashCode()

{

return EditionName.GetHashCode() + ReleseDate.GetHashCode() + PrintEdition.GetHashCode();

}

public override string ToString()

{

return string.Format("Название издания:{0}\n Дата выхода издания: {1},\n Тираж издания: {2}.\n", EditionName, ReleseDate, PrintEdition);

}

}

interface IRateAndCopy

{

double Rating { get; }

object DeepCopy();

}

class Magazine : Edition

{

enum Frequency

{

Weekly,

Monthly,

Yearly

}

Frequency FrequencyOfRelis;

protected ArrayList MagazineEditorList=new ArrayList();

protected ArrayList ListOFArticles = new ArrayList();

Magazine(string MagazineName, Frequency FrequencyOfRelis, DateTime RelissDate, int Edition) : base(MagazineName,RelissDate,Edition)

{

this.FrequencyOfRelis = FrequencyOfRelis;

}

public Magazine():base()

{

this.FrequencyOfRelis = Frequency.Yearly;

}

Frequency FRelise

{

get { return FrequencyOfRelis; }

set { FrequencyOfRelis = value; }

}

double Average

{

get

{

double retng = 0;

foreach (var a in ListOFArticles)

{

retng += (a as Article).Rating;

}

return retng / ListOFArticles.Capacity;

}

}

public ArrayList ArticleList

{

get { return ListOFArticles; }

set { ListOFArticles = value; }

}

public ArrayList MagazinEditorlist

{

get { return MagazinEditorlist; }

}

public void FAddArticles (params Article[] a )

{

ListOFArticles.AddRange(a);

}

public override object DeepCopy()

{

Magazine klon = new Magazine(EditionName, FrequencyOfRelis, ReleseDate, PrintEdition);

return klon;

}

public Edition s

{

get

{

Edition obj = new Edition(EditionName, ReleseDate,PrintEdition);

return obj;

}

}

public override string ToString()

{

string a = string.Empty;

for (int i = 0; i < ListOFArticles.Count; i++)

{

a += (ListOFArticles[i] as Article).ToString() + "##\n-------\n";

}

Console.WriteLine("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

string aa = string.Empty;

for (int i = 0; i < MagazineEditorList.Count; i++)

{

aa += (MagazineEditorList[i] as Person) + "!!!!\n-------\n";

}

return string.Format("\n {3} Частота выхода:{0}," +

"\n Список статей:{1}, \n Список редакторов {2}",

FrequencyOfRelis, a, aa, base.ToString()

);

}

public string ToShortString()

{

return

base.ToString() +

" Частота выхода:" + FrequencyOfRelis + "\n" +

"Средний рейтинг статей:" + Average + "\n";

}

}

class Person

{

string name;

string surname;

DateTime \_DateOfBirth;

public Person(string name, string surname, DateTime DateTime)

{

this.name = name;

this.surname = surname;

this.\_DateOfBirth = DateTime;

}

public Person()

{

this.name = "Нет данных о имени";

this.surname = "Нет данных о фамилии";

this.\_DateOfBirth =new DateTime(2019,01,01);

}

public string PName

{

get { return name; }

set { name = value; }

}

public string PSurname

{

get { return surname; }

set { surname = value; }

}

public System.DateTime PDayOfBirth

{

get { return \_DateOfBirth; }

set { \_DateOfBirth = value; }

}

public int Info

{

get { return \_DateOfBirth.Year; }

set { \_DateOfBirth = new DateTime(); }

}

public static bool operator ==(Person a, Person b)

{

return a.Equals(b);

}

public override bool Equals(object obj)

{

Person person = (Person)obj;

if (this.name == person.name && this.surname == person.surname && this.\_DateOfBirth == person.\_DateOfBirth)

return true;

else

return false;

}

public static bool operator !=(Person a, Person b)

{

return !a.Equals(b);

}

public override int GetHashCode()

{

return name.GetHashCode() + surname.GetHashCode() + \_DateOfBirth.GetHashCode();

}

public override string ToString()

{

return string.Format(" Имя: {0} {3}, \n Фамилия: {1} {4}, \n Год рождения {2} {5}. \n", name, surname, Info, name.GetHashCode(),surname.GetHashCode(), \_DateOfBirth.GetHashCode());

}

public string ToShortString()

{

return " Имя:" + name + "\n" + " Фамилия:" + surname;

}

public virtual object DeepCopy()

{

Person klon = new Person(name, surname, \_DateOfBirth);

return klon;

}

}

public static void Main()

{

DateTime dt1 = new DateTime(2019,10,13);

Edition a1 = new Edition("Издание1", dt1, 200);

Console.WriteLine(a1);

Console.WriteLine("значение хэш - кода: " + a1.GetHashCode());

Edition a2 = new Edition("Издание1", dt1, 200);

Console.WriteLine(a2);

Console.WriteLine("значение хэш - кода: " + a2.GetHashCode()); ;

Console.WriteLine("ссылки на объекты не равны " + (a1 ==a2 ));

Console.WriteLine("объекты равны " + a1.Equals(a2));

Console.WriteLine("Введите тираж издания: ");

DateTime dp3;

dp3 = new DateTime(2019, 09, 20);

Person p3 = new Person("Автор\_нейм", "Автор\_сорнэйм", dp3);

try

{

Edition e = new Edition("trala;la", DateTime.Now, -9);

Console.WriteLine(e);

e.PrintEdition\_ = 0;

}

catch (DivideByZeroException v) { Console.WriteLine(v.Message); }

catch (ArgumentException v) { Console.WriteLine(v.Message); }

DateTime pdt1 = new DateTime(1990, 09, 20);

DateTime pdt2 = new DateTime(1995, 05, 07);

DateTime pdt3 = new DateTime(1991, 02, 08);

Person p1 = new Person("Имя1", " Фамилия1", pdt1);

Person p2 = new Person("Имя2", " Фамилия2", pdt2);

Magazine m = new Magazine();

m.FAddArticles(new Article(p1, "Мышки", 4), new Article(p2, "Кролики", 10.0), new Article(p3, "Статья\_нейм", 7.4));

Console.WriteLine("\*\*\*\n" + m);

Edition m1 = (Edition)m.DeepCopy();

Console.WriteLine(m);

Console.ForegroundColor = ConsoleColor.Cyan;

Console.WriteLine(m1);

Console.ReadKey();

}

}

Результат программы представлен на рисунке 15

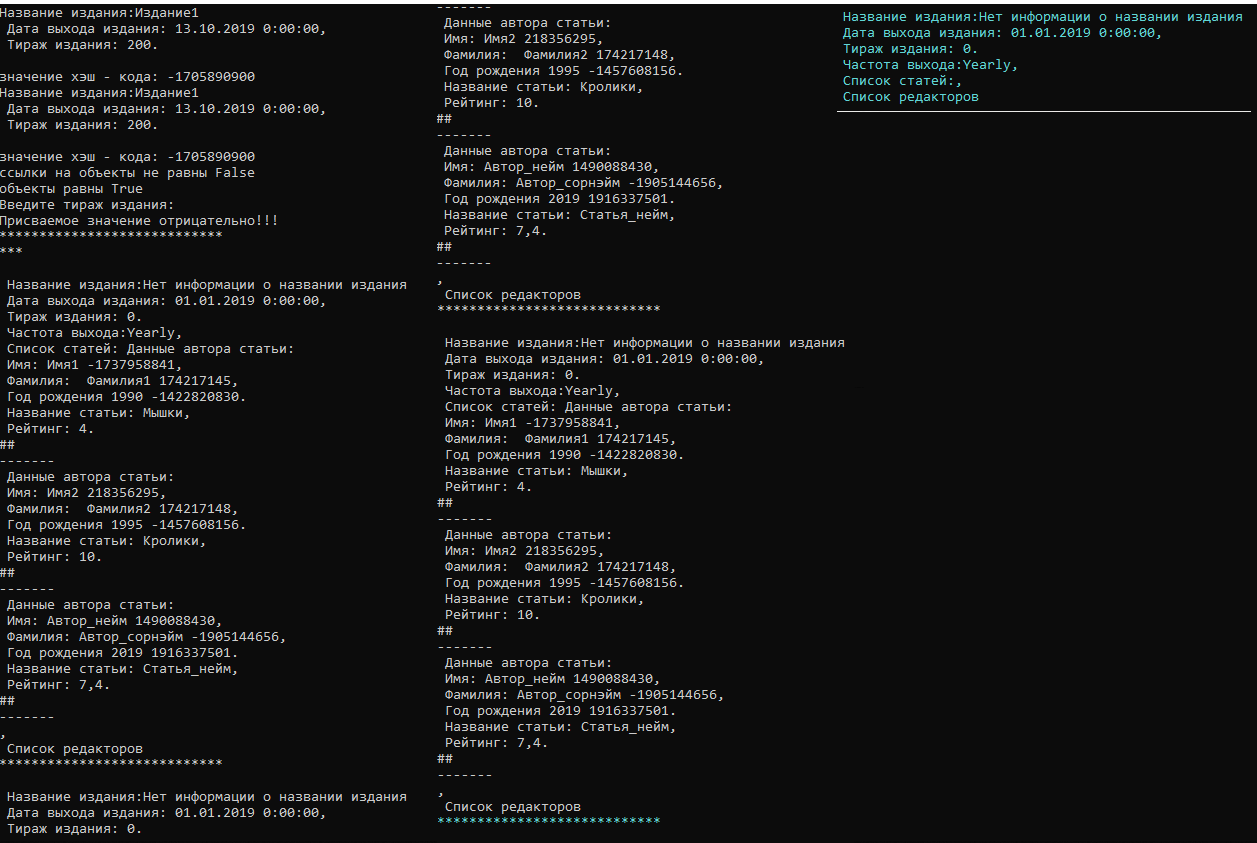


Рисунок 15

# Программирование быстродействующих информационных систем

Создать параметризованную коллекцию для хранения объектов типа:

Разработать приложение для работы с коллекцией, позволяющее выполнять:

* добавление элемента в коллекцию;
* удаление элемента из коллекции;
* сортировку данных по различным критериям;
* вывод элементов;
* поиск элемента по заданному условию;
* считывание данных из XML-файла;
* запись данных в тот же или указанный XML-файл.

class Student

{

public string name;

public string surname;

public DateTime DayOfBirth;

int count = 0;

public Student ()

{

name = "Нет информации о имени";

surname = "Нет информации о фамилии";

DayOfBirth = DateTime.Now;

}

public Student (string name\_,string surname\_, DateTime DayOfBirth\_)

{

name = name\_;

surname = surname\_;

DayOfBirth = DayOfBirth\_;

count++;

}

public void all\_inf ()

{

Console.WriteLine("Имя: {0};\nФамилия: {1};\nДата Рождения: {2}.", name, surname, DayOfBirth);

}

}

static void Main(string[] args)

{

DateTime da1 = new DateTime(2002, 10, 23);

DateTime da2 = new DateTime(2001, 8, 22);

DateTime da3 = new DateTime(2002, 12, 4);

DateTime da4 = new DateTime(2001, 10, 7);

Student a1 = new Student("ИмяУ", "ФамилияО", da1);

Student a2 = new Student("ИмяК", "ФамилияЛ", da2);

Student a3 = new Student("ИмяA", "ФамилияЦ", da3);

Student a4 = new Student("ИмяЩ", "ФамилияМ", da4);

List<Student> all = new List<Student>{ a1, a2, a3, a4 };

Console.WriteLine("Ведите номер опреции: ");

Console.WriteLine("1 Сортировка студентов по имени");

Console.WriteLine("2 Сортировка студентов по фамилии");

Console.WriteLine("3 Сортировка студентов по дате рождения");

Console.WriteLine("4 Поиск по критерию имя");

Console.WriteLine("5 Поиск по критерию фамилия");

Console.WriteLine("6 Поиск по критерию дата рождения");

string buf = Console.ReadLine();

int num = Convert.ToInt32(buf);

switch (num)

{

case 1:

var sort\_name = from u1 in all

orderby u1.name

select u1;

foreach (Student u1 in sort\_name)

{

u1.all\_inf();

}

break;

case 2:

var sort\_surname = from u1 in all

orderby u1.surname

select u1;

foreach (Student u1 in sort\_surname)

{

u1.all\_inf();

}

break;

case 3:

var sort\_date = from u1 in all

orderby u1.DayOfBirth

select u1;

foreach (Student u1 in sort\_date)

{

u1.all\_inf();

}

break;

case 4:

Console.WriteLine("Введите имя: ");

string search\_name = Console.ReadLine();

Student stname = all.Find((x) => x.name == search\_name);

try

{

stname.all\_inf();

}

catch

{

Console.WriteLine("Нет такого имени!");

}

break;

case 5:

Console.WriteLine("Введите фамилию: ");

string search\_surname = Console.ReadLine();

Student stsurname = all.Find((x) => x.surname == search\_surname);

try

{

stsurname.all\_inf();

}

catch

{

Console.WriteLine("Нет такой фамилии!");

}

break;

case 6:

Console.WriteLine("Введите дату рождения: ");

string search\_date = Console.ReadLine();

DateTime dt = Convert.ToDateTime(search\_date);

Student stdt = all.Find((x) => x.DayOfBirth == dt);

try

{

stdt.all\_inf();

}

catch

{

Console.WriteLine("Нет такой даты рождения!");

}

break;

default:

Console.WriteLine("Нет такого номера операции!");

break;

}

Console.WriteLine();

Console.ReadKey();

}

Результата работы программы представлен на рисунке 15

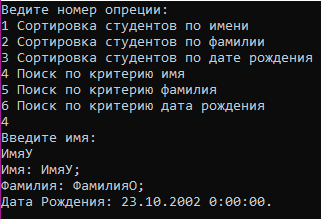


Рисунок 15

# Заключение

За время прохождения учебной практики по программированию были закреплены навыки разработки программ с использованием принципов объектно-ориентированного программирования, продемонстрирована работа с такими элементами как классы, интерфейсы, делегаты, события, коллекции, исследован механизм наследования, продемонстрировано умение создавать LINQ-запросы.

Во время практики удалось накопить неоценимый опыт разработки приложений в среде Microsoft Visual Studio на языке C#, развить умения самостоятельно находить оптимальный метод решения задачи.

Во время практики были закреплены умения

* разрабатывать приложения с использованием объектно-ориентированного подхода к программированию на платформе .NET Framework;
* разрабатывать приложения средствами библиотеки классов Microsoft .NET Framework;
* программировать быстродействующие информационные системы;
* демонстрировать свои проекты.

# Список информационных источников

1. Албахари, Дж. C#6.0. Справочник. Полное описание языка / Дж. Албахари, Б. Албахари. – 6-е изд. – М. : ООО «И.Д. Вильямс», 2016. – 1040 с.: ил.
2. Албахари, Дж. LINQ. Карманный справочник / Дж. Албахари, Б. Албахари. – СПб. : БХВ-Петербург, 2009. – 240 с.: ил.
3. Багласова, Т.Г. Методические указания по оформлению курсовых работ, дипломных проектов и отчетов для учащихся по специальности 2-40 01 01 «Программное обеспечение технологий» / Т.Г. Багласова. – Минск : КБП, 2013. – 29 с.: ил.
4. Гойвертс, Я. Регулярные выражения. Сборник рецептов / Я. Гойвертс, Ст. Левитан. – 2-е изд. – СПб.: Символ-Плюс, 2015. – 704 с.: ил.
5. Дейтел, П. Как программировать на Visual C# 2012 / П. Дейтел, Х. Дейтел. – 5-е изд. – СПб.: Питер, 2014. – 864 с.: ил.
6. Дэвис, А. Асинхронное программирование в C# 5.0 / А. Дэвис. – М.: ДМК Пресс, 2013. – 120 с.: ил.
7. Зиборов, В.В. Visual C# 2012 на примерах / В.В Зиборов. – СПб.: БХВ-Петербург, 2013. – 480 с.: ил.
8. Рихтер, Дж. CLR via C#. Программирование на платформе Microsoft .NET Framework 4.5 на языке C# / Дж. Рихтер. – 4-е изд. – СПб.: Питер, 2013. – 896 с.: ил.
9. Стиллмен, Э. Изучаем C# / Э. Стиллмен, Дж. Грин. – 3-е изд. – СПб.: Питер, 2014. – 816 с.: ил.
10. Троелсен, Э. Язык программирования C# 5.0 и платформа .NET 4.5 / Э. Троелсен. – 6-е изд. – М.: ООО «И.Д. Вильямс», 2013. – 1312 с.: ил.
11. Фицджеральд, М. Регулярные выражения: основы / М. Фицджеральд. – М.: ООО «И.Д. Вильямс», 2015. – 144 с.: ил.
12. Шарп, Джон. Microsoft Visual C#. Подробное руководство / Джон Шарп. – 8-е изд. – – СПб.: Питер, 2017. – 848 с.: ил.
13. Полное руководство по языку программирования С# 8.0 и платформе .NET Core 3 [Электронный ресурс] / METANIT.COM.Сайт о программировании. – metanit.com, 2012-2019. – Режим доступа: http://metanit.com/sharp/tutorial. – Дата доступа: 20.09.2019.
14. Уроки по С# и платформе .NET Framework [Электронный ресурс]/ ProfessorWeb. – Режим доступа: http://professorweb.ru. – Дата доступа: 20.12.2019.
15. Руководство по программированию на языке C# [Электронный ресурс]. – Microsoft.NET, 2019. – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/ programming-guide/. – Дата доступа: 10.10.2019.