**МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ**

**одесский национальный университет имени и.и. мечникова**

**институт математики, экономики и механики**

**КАФЕДРА МАТЕМАТИЧЕСКОГО ОБЕСПЕЧЕНИЯ КОМПЬЮТЕРНЫХ СИСТЕМ**

**Отчет о выполнении летней практики**

|  |  |
| --- | --- |
|  | Студента I курса  Группы 2  Вариант 8  Жужи Георгия Юрьевича  Руководитель: О.А Геренко, И.М Шпинарева  Защищено «\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_201\_ р.  С оценкой \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  Комиссия:  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (ФИО) (Подпись)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (ФИО) (Подпись)  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  (ФИО) (Подпись) |

**Одеса - 2017**

**Задание № 3.2**

**Условие:**

**Члены класса:**

1. Конструктор с одним параметром(строка string с числами, числа в строке разделяются двоеточиями), вычисляющий n – количество чисел в строке, создающий массив из n - элементов и заполняющий его числами из строки. Использовать метод Split Для строк.

1. Конструктор с одним параметром (n), создающий из n элементов и заполняющий его натуральным рядом чисел (1, 2, 3… n), знак числя задается случайным образом;

2. Свойство, доступное только для чтения, для получения максимального

элемента;

3. Метод, выводящий содержимое массива на экран

4. Методы, вычисляющий произведение модулей элементов

расположенных до (левее) первого положительного элемента.

**Код:**

**Класс Massive**

class Massive

{

int[] arr;

int size;

public Massive(string s)

{

string[] s1 = s.Split(':');

this.size = s1.Length;

arr = new int[this.size];

for (int i = 0; i < this.size; i++)

{

arr[i] = Convert.ToInt32(s1[i]);

}

}

public Massive(int n)

{

this.size = n;

arr = new int[this.size];

Random rand = new Random();

for (int i = 0; i < this.size; i++)

{

int t = rand.Next(-50, 50);

if (t >= 0)

{

arr[i] = i;

}

else

{

arr[i] = -i;

}

}

}

public int Max\_new

{

get

{

int max = arr[0];

for (int i = 1; i < this.size; i++)

{

if (max < arr[i])

{

max = arr[i];

}

}

return max;

}

}

public void print()

{

Console.WriteLine();

for (int i = 0; i < this.size; i++)

Console.Write(" {0} ", arr[i]);

Console.WriteLine();

}

public int sum\_mod()

{

int pfe = 0;

int sum = 1;

for (int i = 0; i < this.size - 1; i++)

{

if (this.arr[i] > 0)

{

pfe = i;

break;

}

}

for (int i = pfe - 1; i > 0; i--)

{

sum \*= Math.Abs(arr[i]);

}

return sum;

}

}

**Класс Program**

class Program

{

static void Main(string[] args)

{

Massive rr = new Massive("2:3:4:5");

Massive nn = new Massive(20);

Console.WriteLine("сумма");

Console.WriteLine(nn.sum\_mod());

rr.print();

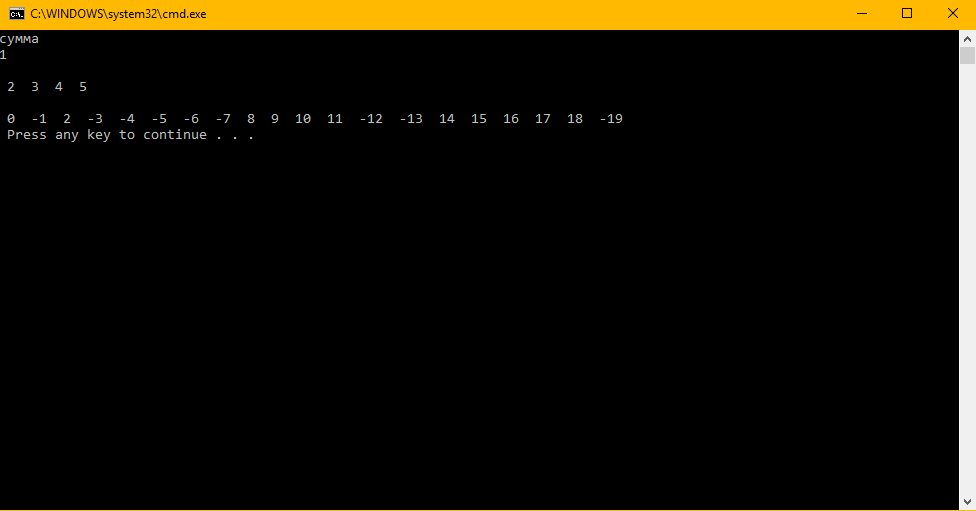
nn.print();

Console.ReadKey();

}

}

**Вывод:**



**Задание № 3.3**

**Условие:**

Проект «Цветик-Семицветик»

**Классы:** Flower (цветок), Boy (мальчик), Girl(девочка) и Program (тестирующий).

В классе возникает два события:

1. Event1 — оторван очередной лепесток

2. Event2 — оторван последний лепесток

В классе Program:

Создать объекты цветок, девочка, мальчик.

В цикле, пока не кончатся лепестки вызывать метод, который уменьшает кол-во лепестков на 1 и генерирует или событие Event1 или Event2.

**1. Класс Flower (цветок)**

**Поля:**

● Открытое статическое поле — сколько осталось лепестков на цветке.

В начале — количество лепестков — 7.

**Методы:**

● Метод генерирующий события: если лепестков больше одного, то кол-во лепестков уменьшается на 1и возникает событие Event1. В противном случае возникает событие Event2.

**2. Класс Girl (девочка)**

Поле массив строк, содержащий 6 желаний девочки.

**Методы:**

● Метод — обрабатывающий события Event1: выводит на экран очередное желание девочки.

● Метод — обработчик события Event2: выводит на экран желание девочки.

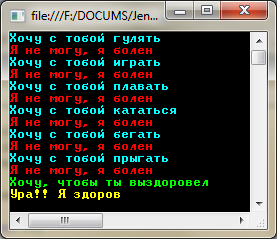
**3. Класс Boy(мальчик)**

**Методы:**

● Метод — обработчик события Event1: выводит на экран сообщение: « Я не могу, я болен».

● Метод — обработчик события Event2: выводит на экран сообщение: «Ура!!! Я здоров».

**Тестирование проекта:**



**Код:**

**Класс Flower:**

class **Flower**

{

public static int flower = 7;

public delegate void MyFlower();

public event MyFlower Event1;

public event MyFlower Event2;

public static int GetFlower { get => flower; set => flower = value; }

public void EvGenerator()

{

if (flower >1 && Event1!=null)

{

flower--;

Event1();

}

else if(Event2 != null)

{

Event2();

}

}

**}**

**Класс Boy:**

class Boy

{

public void Massage1()

{

Console.WriteLine("Я не могу, я болен");

}

public void Massage2()

{

Console.WriteLine("Ура!! Я здоров");

}

}

**Класс Girl:**

**class Girl**

{

private string[] girlWish = { "Хочу, чтобы ты выздоровел", "Хочу с тобой прыгать", "Хочу с тобой бегать", "Хочу с тобой кататься", "Хочу с тобой плавать", "Хочу с тобой играть", "Хочу с тобой гулять"};

//private static int ii = 5;

public string[] GirlWish { get => girlWish; set => girlWish = value; }

public void Wish()

{

//if(Flower.flower>0)

Console.WriteLine(girlWish[Flower.flower]);

}

public void LastWish()

{

Console.WriteLine(girlWish[0]);

}

**}**

**Класс Program:**

class Program

{

static void Main()

{

Flower flowerSevenColor = new Flower();

Girl girl = new Girl();

Boy boy = new Boy();

flowerSevenColor.Event1 += girl.Wish;

flowerSevenColor.Event1 += boy.Massage1;

flowerSevenColor.Event2 += girl.LastWish;

flowerSevenColor.Event2 += boy.Massage2;

for (int i = Flower.flower; i >0; i--)

{

flowerSevenColor.EvGenerator();

}

Console.ReadKey();

}

}

**Вывод:**

****

**Задние № 3.4**

**Условие:**

Класс «Абитуриент» (Порода, кличка, возраст, вес). Должен содержать следующую информацию:

1. Номер экаменационного билета.

2. Фамилия

3. Имя

4. Название факультета

5. Название экзамена

6. Оценка

7. Проходной балл по факультет

Из коллекции класса «Абитуриент» выбираются и выводятся на экран абитуриентов, не поступившие на математический факультет с их суммой баллов.

**Код:**

**Класс Abiturient**

class Abiturient

{

public string name { get; set; }

public string surname { get; set; }

public string exam { get; set; }

public string faculty { get; set; }

public int mark { get; set; }

public int pass\_mark { get; set; }

public Abiturient()

{

surname = "";

faculty = "";

name = "";

exam = "";

mark = 0;

}

public Abiturient(string name, string surname, string exam, string faculty)

{

this.surname = surname;

this.name = name;

this.exam = exam;

this.faculty = faculty;

exam\_num(exam);

name\_fix(name);

surname\_fix(surname);

}

private string exam\_num(string s)

{

if (s.Length == 0)

{

Console.WriteLine("Non-valid name!");

}

return s;

}

public void check\_mark()

{

Console.WriteLine("Mark is: {0}", mark);

}

private string name\_fix(string s)

{

int s1 = s.Length - 1;

if (s[0] == ' ')

s.Remove(0, 1);

if (s1 == ' ')

s.Remove(s1, 1);

return s;

}

private string surname\_fix(string s)

{

if (s[0] == ' ')

s.Remove(0, 1);

if (s.Length - 1 == ' ')

s.Remove(s.Length - 1, 1);

return s;

}

public void print()

{

if (mark < pass\_mark)

{

Console.WriteLine();

Console.WriteLine("Name of student is: {0}", name);

Console.WriteLine("Surname of student is: {0}", surname);

Console.WriteLine("Faculty is: {0}", faculty);

Console.WriteLine("{0} {1} не прошел на факультет из-за низкого балла по {2}, при проходном {3}", name, surname, exam, pass\_mark);

}

}

}

**Класс Program**

class Program

{

static void Main(string[] args)

{

List < Abiturient> ab = new List<Abiturient>();

Random rand = new Random();

string name, surname, exam, faculty;

int mark, pass\_mark = rand.Next(100, 120);

Console.Write("Enter your name: ");

name = Console.ReadLine();

Console.Write("Enter your surname: ");

surname = Console.ReadLine();

Console.WriteLine("What is name of our exam? ");

exam = Console.ReadLine();

Console.Write("Enter your faculty: ");

faculty = Console.ReadLine();

Console.Write("Enter your mark: ");

string s = Console.ReadLine();

mark = Convert.ToInt32(s);

ab.Add(new Abiturient { name = name, surname = surname, exam = exam, faculty = faculty, mark = mark });

foreach (Abiturient xx in ab)

{

if (xx.mark < pass\_mark)

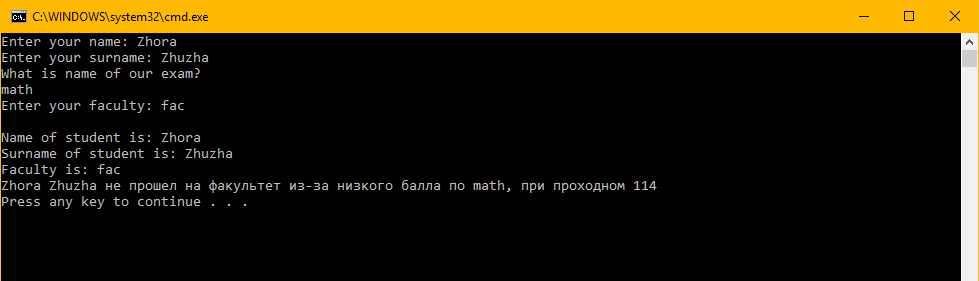
Console.WriteLine("Имя: {0}, Фамилия: {1}, Экзамен: {2}, Факультет: {3}, Оценка: {4}, Проходной: {5}", name, surname, exam, faculty, mark, pass\_mark);

}

}

}

**Вывод:**

****

**Задание № 4.3**

**Условие:**

**Родительский класс - полководец:**

● Поле число битв;

● Поле число побед;

● Метод Q = число побед \* число побед / число битв.

**Дочерний класс:**

● Дополнительное поле Р — число побед с меньшими, чем у противника силами;

● Метод Qp = р\*р/число битв.

**Код:**

**Класс Leader**

class Leader

{

protected int fights;

protected int wins;

protected int q;

protected bool check = true;

public Leader()

{

fights = 0;

wins = 0;

q = 0;

}

public Leader(int fights, int wins)

{

this.fights = fights;

this.wins = wins;

}

public int calc(int fights, int wins)

{

if (fights > 0)

{

int q = wins \* (wins / fights);

}

return q;

}

public void print()

{

if (fights > 0 && fights >= wins)

{

Console.WriteLine("Кол-во побед: {0}", wins);

Console.WriteLine("Кол-во битв: {0}", fights);

}

else

{

Console.WriteLine("\nIncorrect data...");

check = false;

}

}

}

**Класс Slave**

class Slave: Leader

{

private int p;

private int qp = 0;

public Slave(int fights, int wins, int p)

{

this.fights = fights;

this.wins = wins;

this.p = p;

}

public int calc\_new()

{

if (fights > 0)

{

int qp = q + ((p \* p) / fights);

}

return qp;

}

public void print\_new()

{

if (check && p >= wins)

Console.WriteLine("Выиграно с меньшими силами, чем у противника: {0}", p);

else

Console.WriteLine("Seems wrong...\n");

}

}

**Класс Program**

class Program

{

static void Main(string[] args)

{

Random rand = new Random();

for (int i = 0; i < 3; i++)

{

int fights = rand.Next(1, 50);

int wins = rand.Next(0, 50);

int p = rand.Next(1,20);

Slave slave = new Slave(fights, wins, p);

slave.calc(fights, wins);

slave.calc\_new();

slave.print\_new();

slave.print();

}

}

}

**Вывод:**

****

**Задание № 5.1**

**Условие:**

**Создать класс Triangle, описывающий треугольник по трем его сторонам и содержащий:**

● Поле длина первой стороны;

● Поле длина второй стороны;

● Поле длина третьей стороны;

● Конструктор, с тремя параметрами, присваивающий значения этим полям.

● Метод Print – вывод значений полей на экран.

● Метод Perimeter –нахождение периметра треугольника.

● Виртуальный метод Sqr –нахождение площади треугольника по формуле Герона.

● Логический метод Exists – условие существования треугольника.

**Создать дочерний класс Equilateral, описывающий равносторонний треугольник и содержащий:**

● Конструктор. В конструкторе класса использовать один параметр – сторона равностороннего треугольника.

● Переопределяем метод Sqr– вычисление площади равностороннего треугольника по формуле.

● Описать собственный метод Radius, вычисляющий радиус вписанной окружности.

**Код:**

**Класс Triangle**

class Triangle

{

protected int a;

protected int b;

protected int c;

protected int d;

protected double r;

public Triangle()

{

a = 0;

b = 0;

c = 0;

}

public Triangle(int a, int b, int c)

{

this.a = a;

this.b = b;

this.c = c;

}

public int perimeter()

{

return this.d = a + b + c;

}

public virtual double Sqr()

{

double p = this.perimeter() / 2;

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

}

public bool Exists()

{

bool f = true;

if (a >= b + c || b >= a + c || c >= a + b) f = false;

return f;

}

public void print()

{

Console.WriteLine("Первая сторона: {0}, Вторая второна: {1}, Третья сторона: {2}", a, b, c);

Console.WriteLine("Периметр равен: {0}", d);

Console.WriteLine("Площадь: {0}", this.Sqr());

}

}

**Класс Equilateral**

**class Equilateral : Triangle**

{

public int ad;

public Equilateral(int ad) : base(ad, ad, ad)

{

if (Exists() == true)

{

this.ad = ad;

}

else

Console.Write("Cant build the triangle!");

}

public new double Sqr()

{

double s = ((Math.Sqrt(3)) / 4) \* a \* a; return s;

}

public double Radius()

{

return r = a / (2 \* Math.Sqrt(3));

}

public new void print()

{

Console.WriteLine("R = {0}", this.Radius());

}

**}**

**Класс Program**

**class Program**

{

static void Main(string[] args)

{

Random rand = new Random();

string s = Console.ReadLine();

int a = Convert.ToInt32(s);

string s1 = Console.ReadLine();

int b = Convert.ToInt32(s1);

string s2 = Console.ReadLine();

int c = Convert.ToInt32(s2);

Triangle triangle = new Triangle(a, b, c);

triangle.perimeter();

triangle.print();

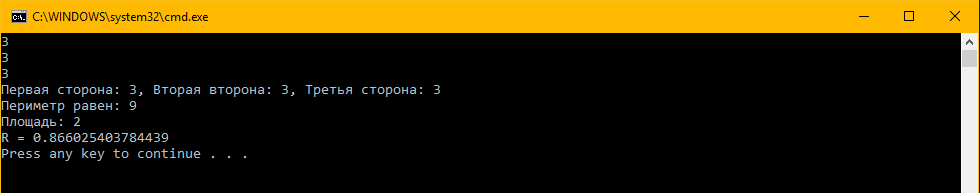
Equilateral eq = new Equilateral(a);

eq.print();

}

**}**

**Вывод:**

****

**Задание № 5.2**

**Условие:**

**Воздать абстрактный класс Student (студень) содержащий:**

**Поля:** фамилия, кол-во посещенных занятий, логическое поле сдал или не сдал зачет.

В классе должны быть абстрактные методы:

● Метод Test (без параметров) — логический метод сдал или не сдал зачет.

● Метод Info — информация (без параметов), который возвращает строку, содержащую ифномацию об объекте.

На его основе реализовать дочерние классы:

● Ordinary – обычный студень;

● Clever – умный студень;

● Genius – гениальный студень.

**Каждый дочерний класс содержит:**

Все поля наследуются из родительского класса.

Конструктор с двумя параметрами: фамилия, количество посещенных занятий.

Переопределить абстрактные методы:

● Test. Заполняет логическое поле:

○ Обычный студент сдает зачет, если посетил 20 занятий.

○ Умный студент сдает зачем, если посетил больше 10 занятий

○ Гениальный студень сдает зачет, если посетил хотя бы одно занятие

● Info формирует строку с значениями полей.

**Код:**

**Класс Student**

class Student

{

protected string name;

protected int num;

protected bool check;

public Student()

{

name = "";

num = 0;

}

public Student(int num, string name)

{

this.num = num;

this.name = name;

}

public virtual bool Test()

{

if (check == true)

Console.WriteLine("Сдал зачет");

else

Console.WriteLine("Не сдал зачет");

return check;

}

public virtual void Info()

{

Console.WriteLine("Имя: {0}", name);

}

}

**Класс Genius**

class Genius : Student

{

public Genius(int num, string name)

: base(num, name) { }

public new bool Test()

{

if (num >= 1)

{

check = true;

}

else

{

if (num < 1)

check = false;

}

return check;

}

public new void Info()

{

Console.WriteLine("Имя: {0}", name);

if (check)

Console.WriteLine("зачет сдал");

else

Console.WriteLine("зачет не сдал");

Console.WriteLine("посетил занятий: {0}", num);

}

}

**Класс Smart**

class Smart : Student

{

public Smart(int num, string name)

: base(num, name) { }

public new bool Test()

{

if (num >= 10)

{

check = true;

}

else

{

if (num < 10)

check = false;

}

return check;

}

public new void Info()

{

Console.WriteLine("Имя: {0}", name);

if (check)

Console.WriteLine("зачет сдал");

else

Console.WriteLine("зачет не сдал");

Console.WriteLine("посетил занятий: {0}", num);

}

}

**Класс Ordinary**

**class Ordinary : Student**

{

public Ordinary(int num, string name)

: base(num, name) { }

public new bool Test()

{

if (num >= 20)

{

check = true;

}

else

{

if (num < 20)

check = false;

}

return check;

}

public new void Info()

{

Console.WriteLine("Имя: {0}", name);

if (check)

Console.WriteLine("зачет сдал");

else

Console.WriteLine("зачет не сдал");

Console.WriteLine("посетил занятий: {0}", num);

}

**}**

**Класс Program**

**class Program**

{

static void Main(string[] args)

{

Ordinary ord = new Ordinary(15, "обычный");

Smart sm = new Smart(10, "умный");

Genius gn = new Genius(1, "гений");

ord.Test();

ord.Info();

Console.WriteLine();

sm.Test();

sm.Info();

Console.WriteLine();

gn.Test();

gn.Info();

Console.WriteLine();

}

**}**

**Вывод:**

****

**Задание № 5.3**

**Условие:**

**Создать интерфейс Itriangle**

В классе должны быть методы:

● Square - вычисления площади,

● Perimeter - вычисления периметра

● Str - возвращающий строку со значениями площади и периметра.

Создать классы, реализующие интерфейс Itriangle.:

● Isosceles равнобедренный треугольник (по двум сторонам).

● Rectangular прямоугольный треугольник (по двум катетам).

● Equilateral равносторонний треугольник (по одной стороне).

**Код:**

**Класс Equilateral**

class Equilateral : ITriangle

{

private double s;

private double a;

private double p;

public Equilateral(double a)

{

this.a = a;

}

public double Sqr()

{

this.s = (a \* a \* Math.Sqrt(3)) / 4;

return s;

}

public double perimeter()

{

this.p = a \* 3;

return p;

}

public void Str()

{

Console.WriteLine("Площадь = {0}", this.s);

Console.WriteLine("Периметр = {0}", this.p);

}

}

**Класс Isosceles**

**class Isosceles : ITriangle**

{

private double s;

private int a;

private int b;

private double p;

public Isosceles(int a, int b)

{

this.b = b;

this.a = a;

}

public double Sqr()

{

this.s = 1 / 4 \* b \* Math.Sqrt((4 \* a \* a) - (b \* b));

return s;

}

public double perimeter()

{

this.p = a + b;

return p;

}

public void Str()

{

Console.WriteLine("Площадь = {0}", this.s);

Console.WriteLine("Периметр = {0}", this.p);

}

**}**

**Класс Rectangular**

**class Rectangular : ITriangle**

{

private double s;

private double a;

private double b;

private double p;

public Rectangular(double a, double b)

{

this.a = a;

this.b = b;

}

public double Sqr()

{

this.s = (a \* b) / 2;

return s;

}

public double perimeter()

{

double c = Math.Sqrt(a \* a + b \* b);

this.p = a + b + c;

return p;

}

public void Str()

{

Console.WriteLine("Площадь = {0}", this.s);

Console.WriteLine("Периметр = {0}", this.p);

}

**}**

**Класс Program**

**class Program**

{

static void Main(string[] args)

{

string s = Console.ReadLine();

int a = Convert.ToInt32(s);

string s1 = Console.ReadLine();

int b = Convert.ToInt32(s1);

Console.WriteLine("Тип треугольника: равносторонний, прямоугольный, равносторонний");

Console.WriteLine();

string s2 = Console.ReadLine();

if (s2 == "равносторонний")

{

ITriangle rs = new Equilateral(a);

rs.perimeter();

rs.Sqr();

rs.Str();

}

else

{

if (s2 == "прямоугольный")

{

ITriangle pg = new Rectangular(a, b);

pg.perimeter();

pg.Sqr();

pg.Str();

}

else

{

if (s2 == "равносторонний")

{

ITriangle rt = new Isosceles(a, b);

rt.perimeter();

rt.Sqr();

rt.Str();

}

}

}

}

}

public interface ITriangle

{

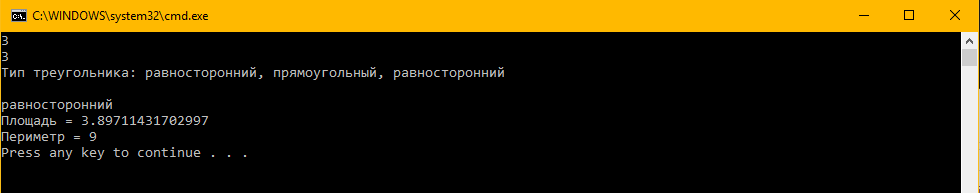
double Sqr();

double perimeter();

void Str();

**}**

**Вывод:**

****

**Задание № 6.3 & Задание № 6.4**

**Условие:**

● comboBox, для выбора буквы из списка.

● Два checkBox, для выбора одного или двух действий.

● Дополнительный элемент label для вывода второго результата

Если на форме есть поле, в которое будет вводиться число, то сделать так, чтобы в это поле нельзя было ввести ничего кроме цифр.

Приложение должно быть спроектировано так, чтобы кнопка ОК была неактивной пока:

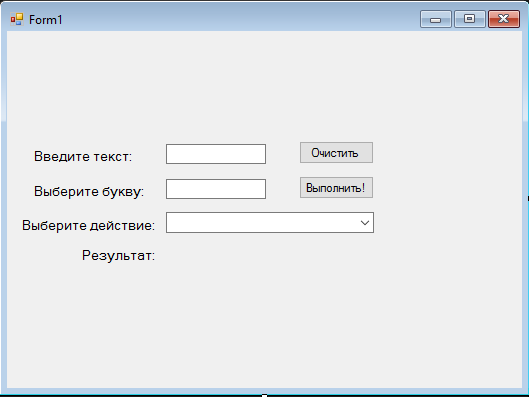
● не выбран символ или число с помощью элементов управления

● не будет заполнено поле, содержащее входную строку,

● не выбрано задание

Кнопка ОК становится активной после заполнения всех полей и выбора всех элементов.

**Форма:**



**Код:**

**private void button1\_Click(object sender, EventArgs e)**

{

string s1 = textBox1.Text;

string s2 = textBox2.Text;

char c = s1[s1.Length-1];

char d = s2[s2.Length - 1];

if (comboBox1.SelectedIndex == 0)

{

if (s1[0] == s2[0])

{

label5.Text = "Da";

}

}

else if (comboBox1.SelectedIndex == 1)

{

if (c == d)

{

label5.Text = "Da";

}

else

label5.Text = "Net";

}

}

private void button2\_Click(object sender, EventArgs e)

{

textBox1.Clear();

textBox2.Clear();

label5.Text = null;

comboBox1.Text = null;

}

private void Form1\_Load(object sender, EventArgs e)

{

button1.Enabled = false;

}

private void textBox2\_KeyPress(object sender, KeyPressEventArgs e)

{

if (!char.IsLetter(e.KeyChar)) e.Handled = true;

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

if (textBox1.TextLength > 0 && textBox2.TextLength > 0 && comboBox1.SelectedIndex >= 0)

{

button1.Enabled = true;

}

else

if (button1.Enabled) button1.Enabled = false;

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

if (textBox1.TextLength > 0 && textBox2.TextLength > 0 && comboBox1.SelectedIndex >= 0)

{

button1.Enabled = true;

}

else

if (button1.Enabled) button1.Enabled = false;

}

private void comboBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

if (textBox1.TextLength > 0 && textBox2.TextLength > 0 && comboBox1.SelectedIndex >= 0)

{

button1.Enabled = true;

}

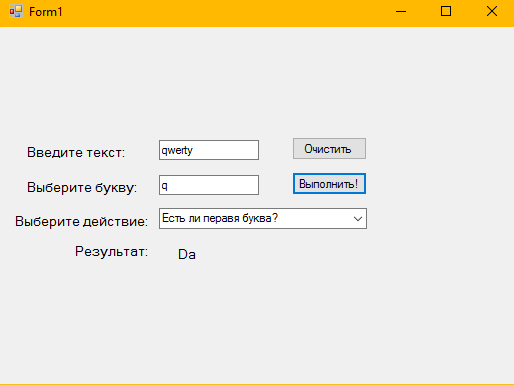
else

if (button1.Enabled) button1.Enabled = false;

}

**}**

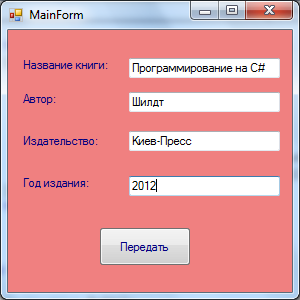
**Вывод:**

****

**Зачетная работа**

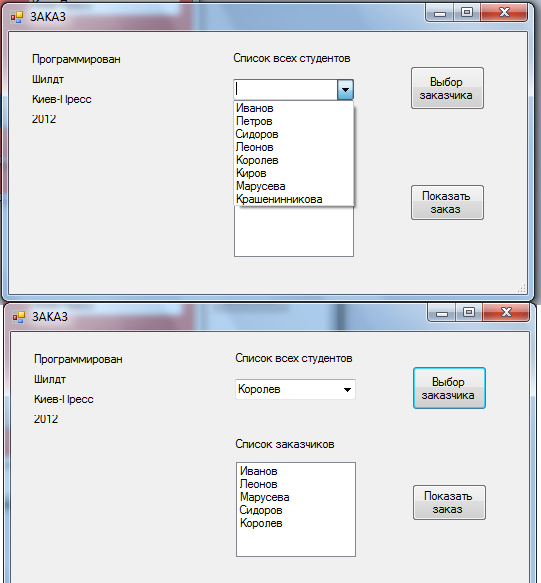
**Условие:**

Надо промоделировать процесс выдачи учебника студентам, выбранным из списка(список уже заполнен). Проект должен состоять из трех форм.

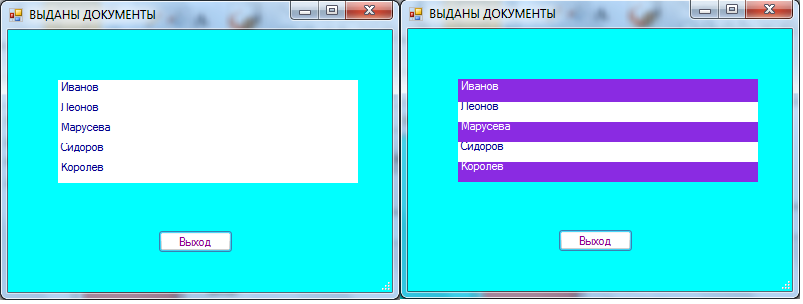


В первом окне задаются данные учебника (надо предусмотреть, что библиотекарь может добавить текстовое поле на форму), нажимается кнопка «ПЕРЕДАТЬ».

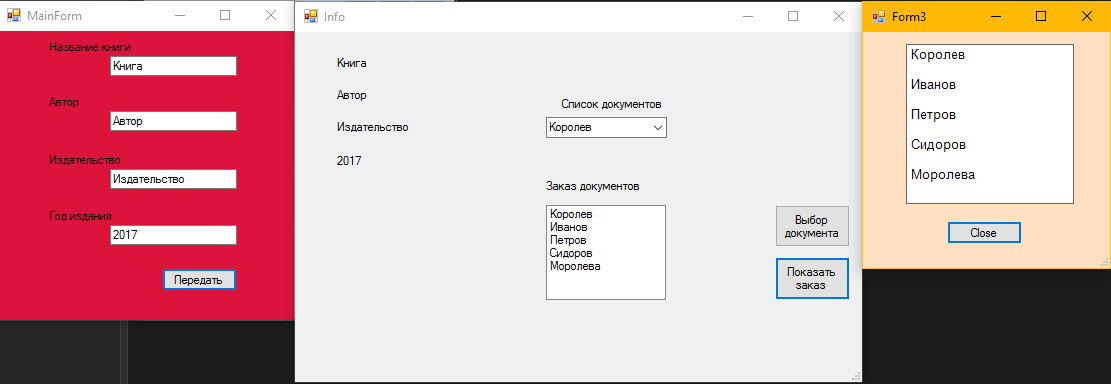
Должно отобразиться второе окно с отображенными данными учебника.



Заказчик из них выбирает студентов, которым нужна эта книга. Список отображается в листбоксе. Затем нажимается кнопка «Показать список».Должно отобразиться третье окно с метками с выбранными фамилиями студентов. При нажатии на метку у нее должен поменяться цвет фона и цвет букв.



**Форма 1 & Форма 2 & форма 3:**

****

**Код:**

**Класс ListStudents**

**class ListStudents**

{

public static List<string> students = new List<string>();

public static void Add(string ss)

{

students.Add(ss);

}

**}**

**Класс User**

**class User**

{

private string id;

private string name;

private string lastName;

private string patr;

public string Id { get => id; set => id = value; }

public string Name { get => name; set => name = value; }

public string LastName { get => lastName; set => lastName = value; }

public string Patr { get => patr; set => patr = value; }

public User(string id, string nm, string ln, string pt)

{

Id = id;

name = nm;

lastName = ln;

Patr = pt;

}

**}**

**Класс Program**

**static class Program**

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

**}**

**Класс Форма 1**

**public partial class Form1 : Form**

{

Form2 form2;

User user;

public Form1()

{

InitializeComponent();

tbId.TextChanged += checkData;

tbLastName.TextChanged += checkData;

tbName.TextChanged += checkData;

tbPatr.TextChanged += checkData;

btTransfer.Enabled = false;

}

private void checkData(object sender, EventArgs e)

{

if (tbId.Text == "")

{

btTransfer.Enabled = false;

return;

}

if (tbLastName.Text == "" || tbName.Text == "" || tbPatr.Text == "")

{

btTransfer.Enabled = false;

return;

}

Func<string, bool> isLetter = s =>

{

foreach (char ch in s)

{

if (!char.IsLetter(ch))

{

return false ;

}

}

return true;

};

Func<string, bool> isDigit = s =>

{

foreach (char ch in s)

{

if (!char.IsDigit(ch))

{

return false;

}

}

return true;

};

bool bl = isLetter(tbName.Text);

if(!bl)

{

btTransfer.Enabled = false;

return;

}

bl = isLetter(tbLastName.Text);

if (!bl)

{

btTransfer.Enabled = false;

return;

}

bl = isDigit(tbPatr.Text);

if (!bl)

{

btTransfer.Enabled = false;

return;

}

btTransfer.Enabled = true;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void btTransfer\_Click(object sender, EventArgs e)

{

form2 = new Form2();

form2.CoWorker = new User(tbId.Text, tbName.Text, tbLastName.Text, tbPatr.Text);

form2.ShowDialog();

}

private void Form1\_Click(object sender, EventArgs e)

{

this.BackColor = System.Drawing.Color.Crimson;

}

**}**

**Класс Форма 2**

**public partial class Form2 : Form**

{

User user;

Form3 form3;

public Form2()

{

InitializeComponent();

}

internal User CoWorker { get => user; set => user = value; }

private void button2\_Click(object sender, EventArgs e)

{

form3 = new Form3();

form3.ShowDialog();

}

private void Form2\_Load(object sender, EventArgs e)

{

lbId.Text = user.Id;

lbLastName.Text = user.LastName;

lbName.Text = user.Name;

lbPatr.Text = user.Patr;

cmbSelect.Items.Add("Королев");

cmbSelect.Items.Add("Иванов");

cmbSelect.Items.Add("Петров");

cmbSelect.Items.Add("Сидоров");

cmbSelect.Items.Add("Королев");

cmbSelect.Items.Add("Моролева");

cmbSelect.SelectedIndex = 0;

}

private void button1\_Click(object sender, EventArgs e)

{

string selectDoc = cmbSelect.SelectedItem.ToString();

listBox.Items.Add(selectDoc);

ListStudents.Add(selectDoc);

}

private void pictureBox1\_Click(object sender, EventArgs e)

{

}

private void label5\_Click(object sender, EventArgs e)

{

}

**}**

**Класс Форма 3**

**public partial class Form3 : Form**

{

public Form3()

{

InitializeComponent();

lbLast.View = View.Tile;

lbLast.OwnerDraw = true;

lbLast.DrawItem += new DrawListViewItemEventHandler(lbLast\_DrawItem);

}

private void Form3\_Load(object sender, EventArgs e)

{

foreach (string str in ListStudents.students)

{

lbLast.Items.Add(str);

}

}

private void button1\_Click(object sender, EventArgs e)

{

this.Close();

}

private void lbLast\_DrawItem(object sender, DrawListViewItemEventArgs e)

{

if ((e.State & ListViewItemStates.Selected) != 0)

{

String str = e.Item.Text;

e.Graphics.FillRectangle(Brushes.Purple, e.Bounds);

e.Graphics.DrawString(str, new Font("Arial", 10), Brushes.White, new Point(e.Item.Position.X, e.Item.Position.Y));

}

else

{

String str = e.Item.Text;

e.Graphics.FillRectangle(Brushes.White, e.Bounds);

e.Graphics.DrawString(str, new Font("Arial", 10), Brushes.Black, new Point(e.Item.Position.X, e.Item.Position.Y));

}

}

**}**