

Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по учебной практике

УП 01.01

Выполнил: Малюгин В.А.

Группа: ПР-22

Преподаватель: Мирошниченко Г.В.

2023

Содержание

[1. Задание №3 Индивидуальное задание (вариант 4) 3](#_Toc138084566)

[**1.1** **Описание задачи** 3](#_Toc138084567)

[**1.2** **Структура проекта** 3](#_Toc138084568)

[**Computer.cs** 3](#_Toc138084569)

[**PotomocComputer.cs** 5](#_Toc138084570)

[**Form1.cs** 6](#_Toc138084571)

[**UnitTest1.cs** 10](#_Toc138084572)

[**1.3 Описание разработанных функций** 19](#_Toc138084573)

[**1.4 Алгоритм решения** 20](#_Toc138084574)

[**1.5 Используемые библиотеки** 23](#_Toc138084575)

[**1.6 Тестовые случаи** 24](#_Toc138084576)

[**1.7 Используемые инструменты** 26](#_Toc138084577)

[**1.8 Описание пользовательского интерфейса** 26](#_Toc138084578)

[**1.9 Приложение (pr screen экранов)** 27](#_Toc138084579)

# 1. Задание №3 Индивидуальное задание (вариант 4)

## **Описание задачи**

Создать проект для демонстрации работы: ввод-вывод информации об

объектах базового класса и класса потомка

Компьютер:

- наименование процессора;

-тактовая частота процессора (МГц);

- объем оперативной памяти

(Мб). Q = (0,3·частота) +

память)

Класс потомок:

P: объём винчестера (Гб)

Qp=Q+0,5·Р

## **Структура проекта**

### **Computer.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace zadanie3

{

public class Computer

{

public string nameProz;

public double chastora;

public int opMemory;

public int price;

public int reiting;

public Computer(string nameProz, double chastora, int opMemory, int price, int reiting)

{

this.nameProz = nameProz;

this.chastora = chastora;

this.opMemory = opMemory;

this.price = price;

this.reiting = reiting;

}

public Computer(double chastora, int opMemory, int price, int reiting, Dictionary<string, double> valueQ)

{

int i = 1;

string text = "не указано ";

string name = text;

while (true)

{

bool repeat = valueQ.Any(s => s.Key == name);

if (repeat == false) break;

else name = text + i;

i++;

}

nameProz = name;

this.chastora = chastora;

this.opMemory = opMemory;

this.price = price;

this.reiting = reiting;

}

public double Q(double chastota, int opMemory) //считает q

{

double q = (0.3 \* chastota) + opMemory;

return q;

}

static public void Print(DataGridView table, List<Computer> computers) //вывод в датагрид

{

table.Rows.Clear();

int i = 0;

foreach (var text in computers)

{

table.Rows.Add();

table[0, i].Value = text.nameProz;

table[1, i].Value = text.chastora;

table[2, i].Value = text.opMemory;

table[3, i].Value = text.price;

table[4, i].Value = text.reiting;

i++;

}

}

static public void Print(ListBox list, Dictionary<string, double> valueQ, string name) //вывод в листбокс

{

list.Items.Clear();

int i = 0;

foreach (var text in valueQ)

{

list.Items.Add($"Название: "+ text.Key +" " + name + " = " + text.Value);

i++;

}

}

public void Add(List<Computer> computers) //добавление в лист

{

computers.Add(this);

}

public void Add(Dictionary<string, double> valueQ, List<Computer> list) //добавление в словарь

{

valueQ.Clear();

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

{

valueQ.Add(list[i].nameProz, Q(list[i].chastora, list[i].opMemory));

}

}

static public void Delete(int index, List<Computer> computers) //удаление определеного элемента из листа

{

computers.RemoveAt(index);

}

static public void Delete(int index, List<Computer> computers, Dictionary<string, double> valueQ) //удаление определеного элемента из словаря

{

valueQ.Remove(computers[index].nameProz);

}

static public void Delete(List<Computer> computers, Dictionary<string, double> valueQ) //удаление всего

{

computers.Clear();

valueQ.Clear();

}

}

}

### **PotomocComputer.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace zadanie3

{

public class PotomocComputer : Computer

{

public int valueP;

public PotomocComputer(string nameProz, double chastora, int opMemory, int price, int reiting, int valueP) :

base(nameProz, chastora, opMemory, price, reiting)

{

this.valueP = valueP;

}

public PotomocComputer(double chastora, int opMemory, int price, int reiting, Dictionary<string, double> valueQ, int valueP)

: base(chastora, opMemory, price, reiting, valueQ)

{

this.valueP = valueP;

}

public void Add(List<PotomocComputer> computers) //добавление в лист

{

computers.Add(this);

}

static public void Delete(int index, List<PotomocComputer> computers) //удаление определеного элемента из листа

{

computers.RemoveAt(index);

}

static public void Delete(int index, List<PotomocComputer> computers, Dictionary<string, double> valueP) //удаление определеного элемента из словаря

{

valueP.Remove(computers[index].nameProz);

}

static public void Delete(List<PotomocComputer> computers, Dictionary<string, double> valueQ) //удаление всего

{

computers.Clear();

valueQ.Clear();

}

static public void Print(DataGridView table, List<PotomocComputer> computers) //вывод в датагрид

{

table.Rows.Clear();

int i = 0;

foreach (var text in computers)

{

table.Rows.Add();

table[0, i].Value = text.nameProz;

table[1, i].Value = text.chastora;

table[2, i].Value = text.opMemory;

table[3, i].Value = text.price;

table[4, i].Value = text.reiting;

table[5, i].Value = text.valueP;

i++;

}

}

public double Qp(int index, List<PotomocComputer> list) //считает q

{

double qp = Q(list[index].chastora, list[index].opMemory) + 0.5 \* list[index].valueP;

return qp;

}

public void Add(Dictionary<string, double> valueP,List<PotomocComputer> list) //добавление в словарь

{

valueP.Clear();

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

{

valueP.Add(list[i].nameProz, Qp(i, list));

}

}

static public Dictionary<string, double> searchPrice(List<PotomocComputer> list, int stoimost)

{

Dictionary<string, double> price = new Dictionary<string, double>();

var potomocComputers = list.Where(predicate: x => x.price < stoimost);

foreach (var text in potomocComputers)

{

price.Add(text.nameProz, text.price);

}

return price;

}

static public Dictionary<string, double> sortMemory(List<PotomocComputer> list)

{

Dictionary<string, double> price = new Dictionary<string, double>();

var potomocComputers = list.OrderByDescending(x => x.opMemory);

foreach (var text in potomocComputers)

{

price.Add(text.nameProz, text.opMemory);

}

return price;

}

}

}

### **Form1.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Windows.Forms;

namespace zadanie3

{

public partial class Form1 : Form

{

public List<Computer> list = new List<Computer>();

public Dictionary<string, double> valueQ = new Dictionary<string, double>();

public List<PotomocComputer> potomoclist = new List<PotomocComputer>();

public Dictionary<string, double> valueP = new Dictionary<string, double>();

public Form1()

{

InitializeComponent();

}

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

{

Delete.Enabled = false;

dataGridView1.AllowUserToAddRows = false;

dataGridView2.AllowUserToAddRows = false;

listBox1.SelectionMode = 0;

listBox2.SelectionMode = 0;

listBox3.SelectionMode = 0;

listBox4.SelectionMode = 0;

numericUpDown1.AllowDrop = false;

naslDel.Enabled = false;

}

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

{

addForm();

}

public string addForm()

{

string text = "";

bool repeat = valueQ.Any(s => s.Key == textBox1.Text);

if (repeat)

{

text = "Такой название процессора уже есть";

MessageBox.Show("Такой название процессора уже есть");

}

else

{

if (textBox1.Text.Length == 0)

{

Computer computer = new Computer((double)numericUpDown1.Value,

(int)numericUpDown2.Value, (int)numericUpDown3.Value, (int)numericUpDown4.Value, valueQ);

computer.Add(list);

computer.Add(valueQ, list);

}

else

{

Computer computer = new Computer(textBox1.Text, (double)numericUpDown1.Value,

(int)numericUpDown2.Value, (int)numericUpDown3.Value, (int)numericUpDown4.Value);

computer.Add(list);

computer.Add(valueQ, list);

}

Computer.Print(dataGridView1, list);

Computer.Print(listBox1, valueQ, "Q");

}

return text;

}

private void dataGridView1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

Delete.Enabled = true;

}

private void dataGridView1\_CellEndEdit(object sender, DataGridViewCellEventArgs e)

{

}

private void tabPage1\_Click(object sender, EventArgs e)

{

}

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

{

Computer.Delete((int)dataGridView1.CurrentCell.RowIndex, list, valueQ);

Computer.Delete((int)dataGridView1.CurrentCell.RowIndex, list);

Computer.Print(dataGridView1, list);

Computer.Print(listBox1, valueQ, "Q");

Delete.Enabled = false;

}

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

{

Delete.Enabled = false;

}

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

{

Computer.Delete(list, valueQ);

Computer.Print(dataGridView1, list);

Computer.Print(listBox1, valueQ, "Q");

}

private void dataGridView1\_CellContentClick(object sender, DataGridViewCellEventArgs e)

{

}

private void listBox1\_SelectedIndexChanged(object sender, EventArgs e)

{

}

private void cheakNumeric(object sender, EventArgs e)

{

if (numericUpDown1.Text == "")

numericUpDown1.Text = numericUpDown1.Value.ToString();

if (numericUpDown2.Text == "")

numericUpDown2.Text = numericUpDown2.Value.ToString();

if (numericUpDown3.Text == "")

numericUpDown3.Text = numericUpDown3.Value.ToString();

if (numericUpDown4.Text == "")

numericUpDown4.Text = numericUpDown4.Value.ToString();

if (numericUpDown5.Text == "")

numericUpDown5.Text = numericUpDown5.Value.ToString();

if (numericUpDown6.Text == "")

numericUpDown6.Text = numericUpDown6.Value.ToString();

if (numericUpDown7.Text == "")

numericUpDown7.Text = numericUpDown7.Value.ToString();

if (numericUpDown8.Text == "")

numericUpDown8.Text = numericUpDown8.Value.ToString();

if (numericUpDown9.Text == "")

numericUpDown9.Text = numericUpDown9.Value.ToString();

}

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

{

bool repeat = valueP.Any(s => s.Key == textBox2.Text);

if (repeat)

{

MessageBox.Show("Такой название процессора уже есть");

}

else

{

if (textBox2.Text.Length == 0)

{

PotomocComputer computer = new PotomocComputer((double)numericUpDown5.Value,

(int)numericUpDown7.Value, (int)numericUpDown8.Value, (int)numericUpDown6.Value, valueP, (int)numericUpDown9.Value);

computer.Add(potomoclist);

computer.Add(valueP,potomoclist);

}

else

{

PotomocComputer computer = new PotomocComputer(textBox2.Text, (double)numericUpDown5.Value,

(int)numericUpDown7.Value, (int)numericUpDown8.Value, (int)numericUpDown6.Value, (int)numericUpDown9.Value);

computer.Add(potomoclist);

computer.Add(valueP,potomoclist);

}

PotomocComputer.Print(dataGridView2, potomoclist);

Computer.Print(listBox3, PotomocComputer.searchPrice(potomoclist, 5000), "Цена: ");

Computer.Print(listBox4, PotomocComputer.sortMemory(potomoclist), "Память: ");

Computer.Print(listBox2, valueP, "P");

}

}

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

{

PotomocComputer.Delete(potomoclist, valueP);

PotomocComputer.Print(dataGridView2, potomoclist);

Computer.Print(listBox3, PotomocComputer.searchPrice(potomoclist, 5000), "Цена: ");

Computer.Print(listBox4, PotomocComputer.sortMemory(potomoclist), "Память: ");

Computer.Print(listBox2, valueP, "P");

}

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

{

PotomocComputer.Delete((int)dataGridView2.CurrentCell.RowIndex, potomoclist, valueP);

PotomocComputer.Delete((int)dataGridView2.CurrentCell.RowIndex, potomoclist);

Computer.Print(listBox3, PotomocComputer.searchPrice(potomoclist, 5000), "Цена: ");

PotomocComputer.Print(dataGridView2, potomoclist);

Computer.Print(listBox2, valueP, "P");

Computer.Print(listBox4, PotomocComputer.sortMemory(potomoclist), "Память: ");

Delete.Enabled = false;

}

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

{

}

private void dataGridView2\_CellClick (object sender, DataGridViewCellEventArgs e)

{

naslDel.Enabled = true;

}

}

}

**UnitTest1.cs**

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System.Collections.Generic;

using zadanie3;

namespace test

{

[TestClass]

public class testQ

{

[TestMethod]

public void q8\_6 ()

{

// исходные данные

string nameProz = "1";

double chastota = 2;

int opMemory = 8;

int price = 1000;

int reiting = 2;

//ожидаемый результат

double expected = 8.6;

// получение значения с помощью тестируемого метода

Computer computer = new Computer(nameProz, chastota, opMemory, price, reiting);

double actual = computer.Q(chastota, opMemory);

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void q11\_5 ()

{

// исходные данные

string nameProz = "1";

double chastota = 5;

int opMemory = 10;

int price = 1000;

int reiting = 2;

//ожидаемый результат

double expected = 11.5;

// получение значения с помощью тестируемого метода

Computer computer = new Computer(nameProz, chastota, opMemory, price, reiting);

double actual = computer.Q(chastota, opMemory);

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void q12\_9 ()

{

// исходные данные

string nameProz = "1";

double chastota = 3;

int opMemory = 12;

int price = 1000;

int reiting = 2;

//ожидаемый результат

double expected = 12.9;

// получение значения с помощью тестируемого метода

Computer computer = new Computer(nameProz, chastota, opMemory, price, reiting);

double actual = computer.Q(chastota, opMemory);

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void q25\_8 ()

{

// исходные данные

string nameProz = "1";

double chastota = 6;

int opMemory = 24;

int price = 1000;

int reiting = 2;

//ожидаемый результат

double expected = 25.8;

// получение значения с помощью тестируемого метода

Computer computer = new Computer(nameProz, chastota, opMemory, price, reiting);

double actual = computer.Q(chastota, opMemory);

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

}

[TestClass]

public class Add

{

[TestMethod]

public void add1 ()

{

// исходные данные

string nameProz = "1";

double chastota = 2;

int opMemory = 8;

int price = 1000;

int reiting = 2;

List<Computer> list = new List<Computer>();

//ожидаемый результат

int expected = 1;

// получение значения с помощью тестируемого метода

Computer computer = new Computer(nameProz, chastota, opMemory, price, reiting);

computer.Add(list);

int actual = list.Count;

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void add2 ()

{

// исходные данные

double chastota = 2;

int opMemory = 8;

int price = 1000;

int reiting = 2;

List<Computer> list = new List<Computer>();

Dictionary<string, double> valueQ = new Dictionary<string, double>();

//ожидаемый результат

int expected = 1;

// получение значения с помощью тестируемого метода

Computer computer = new Computer(chastota, opMemory, price, reiting, valueQ);

computer.Add(list);

int actual = list.Count;

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

}

[TestClass]

public class Del

{

[TestMethod]

public void del1 ()

{

// исходные данные

string nameProz = "1";

double chastota = 2;

int opMemory = 8;

int price = 1000;

int reiting = 2;

List<Computer> list = new List<Computer>();

//ожидаемый результат

int expected = 0;

// получение значения с помощью тестируемого метода

Computer computer = new Computer(nameProz, chastota, opMemory, price, reiting);

computer.Add(list);

Computer.Delete(0, list);

int actual = list.Count;

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void del2 ()

{

// исходные данные

double chastota = 2;

int opMemory = 8;

int price = 1000;

int reiting = 2;

List<Computer> list = new List<Computer>();

Dictionary<string, double> valueQ = new Dictionary<string, double>();

//ожидаемый результат

int expected = 0;

// получение значения с помощью тестируемого метода

Computer computer = new Computer(chastota, opMemory, price, reiting, valueQ);

computer.Add(list);

Computer.Delete(0, list);

int actual = list.Count;

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void del3 ()

{

// исходные данные

double chastota = 2;

int opMemory = 8;

int price = 1000;

int reiting = 2;

int p = 500;

List<PotomocComputer> list = new List<PotomocComputer>();

Dictionary<string, double> valueP = new Dictionary<string, double>();

//ожидаемый результат

int expected = 0;

// получение значения с помощью тестируемого метода

PotomocComputer computer = new PotomocComputer(chastota, opMemory, price, reiting, valueP, p);

computer.Add(list);

PotomocComputer.Delete(0, list);

int actual = list.Count;

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void del4 ()

{

// исходные данные

string nameProz = "1";

double chastota = 2;

int opMemory = 8;

int price = 1000;

int reiting = 2;

int p = 500;

List<PotomocComputer> list = new List<PotomocComputer>();

Dictionary<string, double> valueP = new Dictionary<string, double>();

//ожидаемый результат

int expected = 0;

// получение значения с помощью тестируемого метода

PotomocComputer computer = new PotomocComputer(nameProz, chastota, opMemory, price, reiting, p);

computer.Add(list);

PotomocComputer.Delete(0, list);

int actual = list.Count;

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

}

[TestClass]

public class valueQp

{

[TestMethod]

public void qp258\_6()

{

string nameProz = "1";

double chastota = 2;

int opMemory = 8;

int price = 1000;

int reiting = 2;

int p = 500;

List<PotomocComputer> list = new List<PotomocComputer>();

//ожидаемый результат

double expected = 258.6;

// получение значения с помощью тестируемого метода

PotomocComputer computer = new PotomocComputer(nameProz, chastota, opMemory, price, reiting, p);

computer.Add(list);

double actual = computer.Qp(0, list);

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void qp508\_6 ()

{

string nameProz = "1";

double chastota = 2;

int opMemory = 8;

int price = 1000;

int reiting = 2;

int p = 1000;

List<PotomocComputer> list = new List<PotomocComputer>();

//ожидаемый результат

double expected = 508.6;

// получение значения с помощью тестируемого метода

PotomocComputer computer = new PotomocComputer(nameProz, chastota, opMemory, price, reiting, p);

computer.Add(list);

double actual = computer.Qp(0, list);

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void qp509\_2 ()

{

string nameProz = "1";

double chastota = 4;

int opMemory = 8;

int price = 1000;

int reiting = 2;

int p = 1000;

List<PotomocComputer> list = new List<PotomocComputer>();

double expected = 509.2;

PotomocComputer computer = new PotomocComputer(nameProz, chastota, opMemory, price, reiting, p);

computer.Add(list);

double actual = computer.Qp(0, list);

Assert.AreEqual(expected, actual);

}

[TestMethod]

public void qp505\_2 ()

{

string nameProz = "1";

double chastota = 4;

int opMemory = 4;

int price = 1000;

int reiting = 2;

int p = 1000;

List<PotomocComputer> list = new List<PotomocComputer>();

//ожидаемый результат

double expected = 505.2;

// получение значения с помощью тестируемого метода

PotomocComputer computer = new PotomocComputer(nameProz, chastota, opMemory, price, reiting, p);

computer.Add(list);

double actual = computer.Qp(0, list);

// сравнение ожидаемого результата с полученным

Assert.AreEqual(expected, actual);

}

}

}

## **1.3 Описание разработанных функций**

Q(double chastota, int opMemory)– возвращает q по условию задания в базовом классе computer.

Print(DataGridView table, List<Computer> computers) – выводит лист в datagridview

Print(ListBox list, Dictionary<string, double> valueQ, string name) – вывод словаря в listbox

Add(List<Computer> computers) – добавление информации в лист

Delete(int index, List<Computer> computers) - удаление определеного элемента из листа

Delete(int index, List<Computer> computers, Dictionary<string, double> valueQ) - удаление определеного элемента из словаря

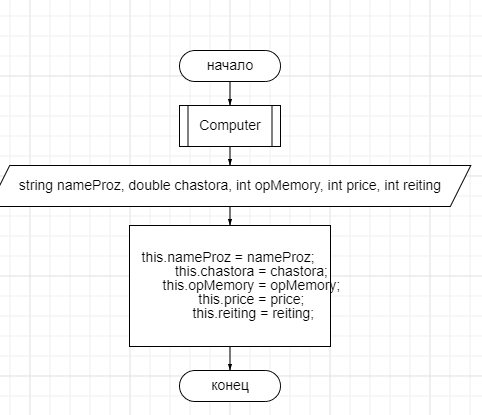
Delete(List<Computer> computers, Dictionary<string, double> valueQ) //отчистка словаря и листа

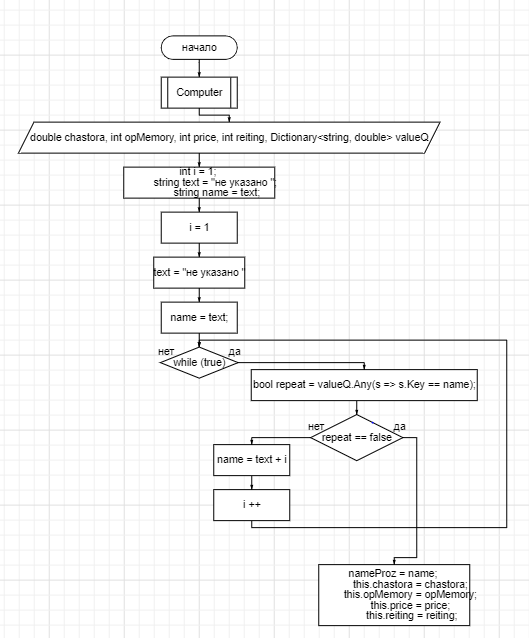
Qp(int index, List<PotomocComputer> list) - возвращает qp по условию задания в классе наследнике PotomocComputer.

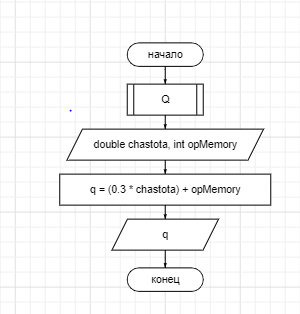
Dictionary<string, double> searchPrice(List<PotomocComputer> list, int stoimost) – находит процессоры дешевле заданной цены

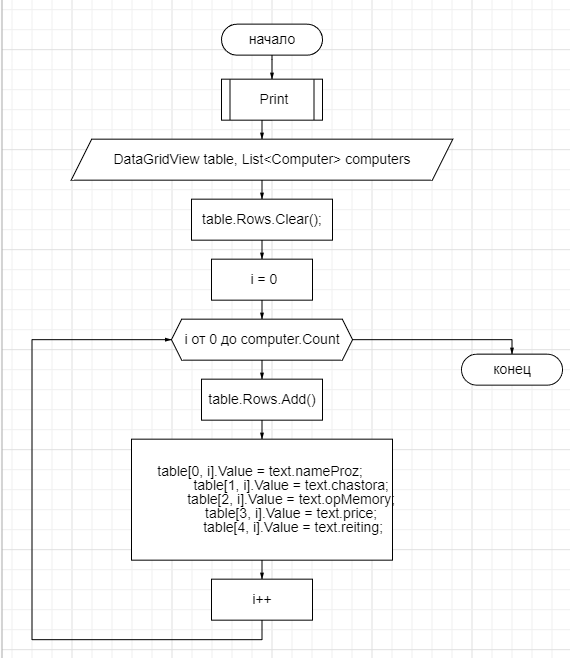
Dictionary<string, double> sortMemory(List<PotomocComputer> list) – сортирует по убыванию процессоры по оперативной памяти.

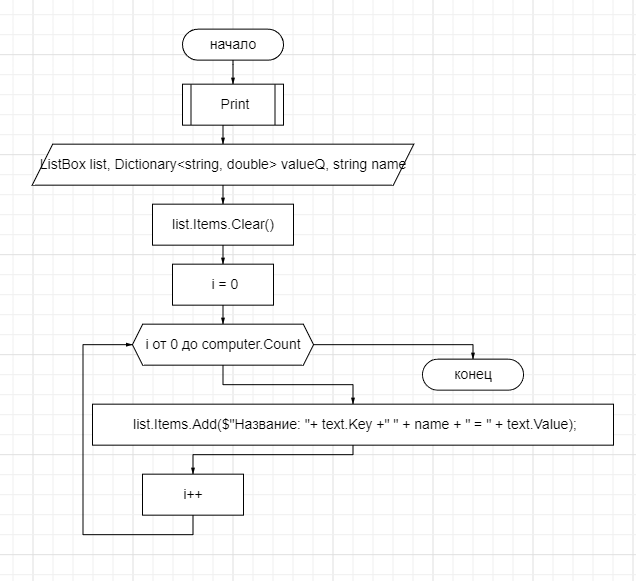
## **1.4 Алгоритм решения**

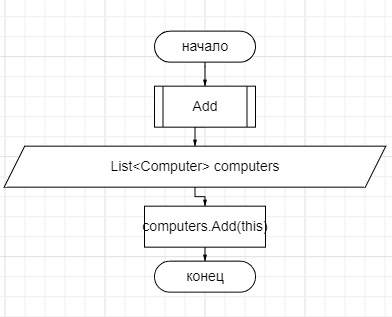


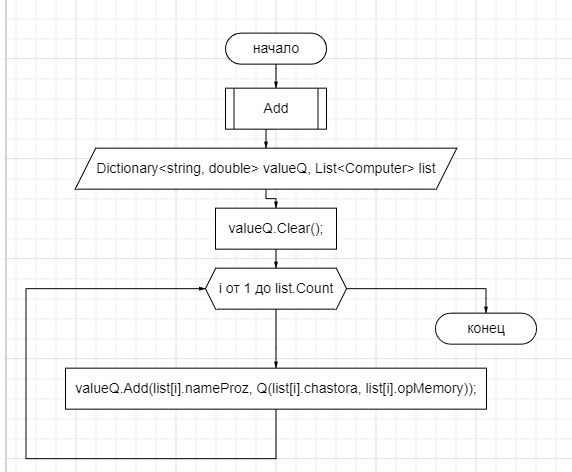


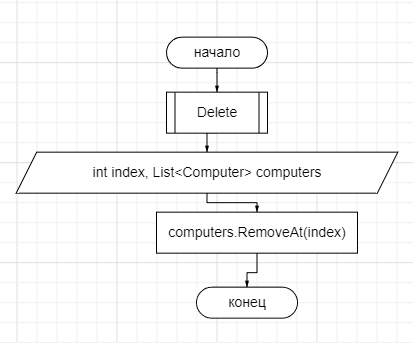


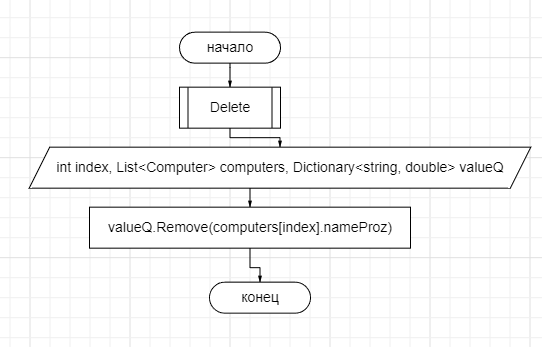


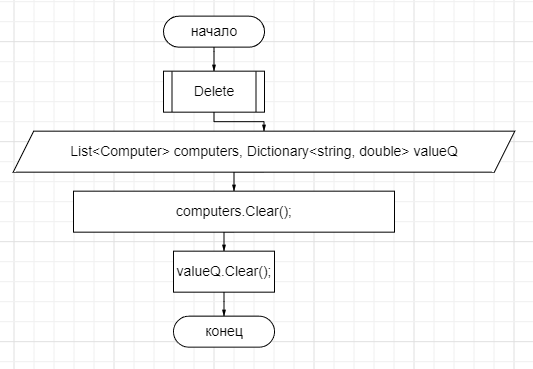


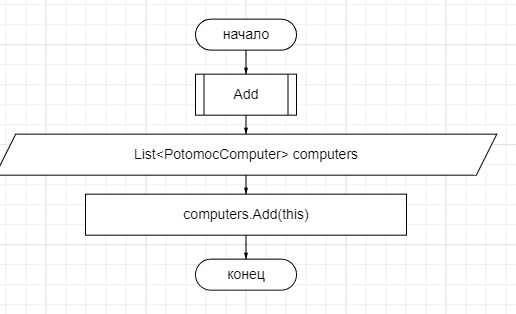
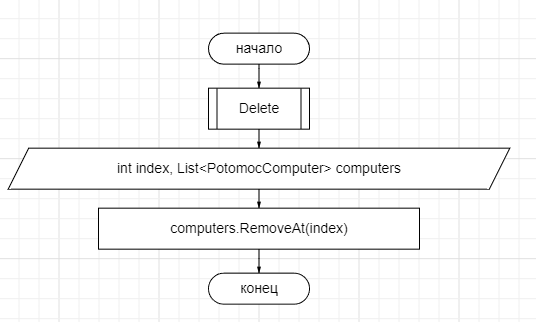


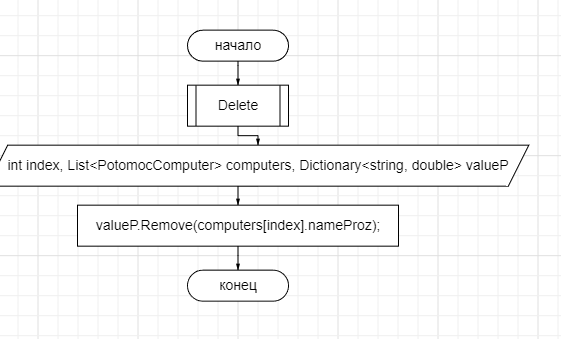


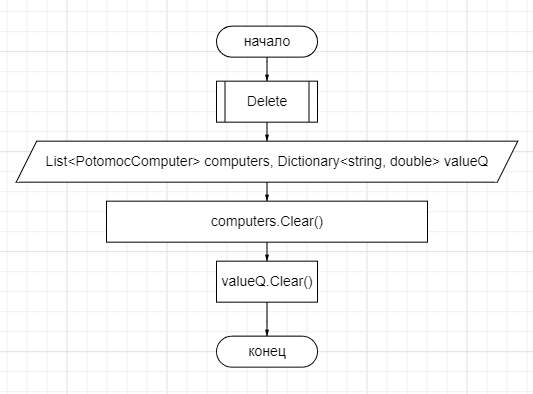


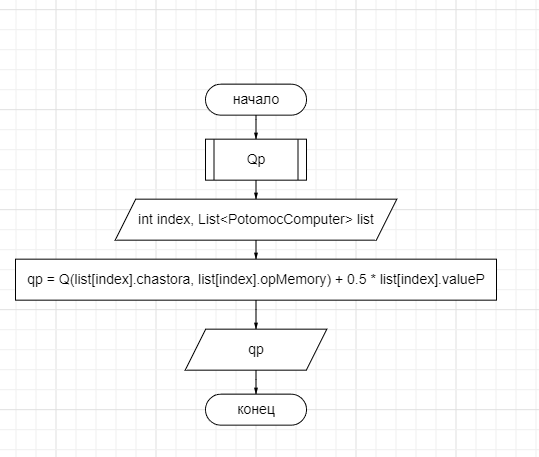


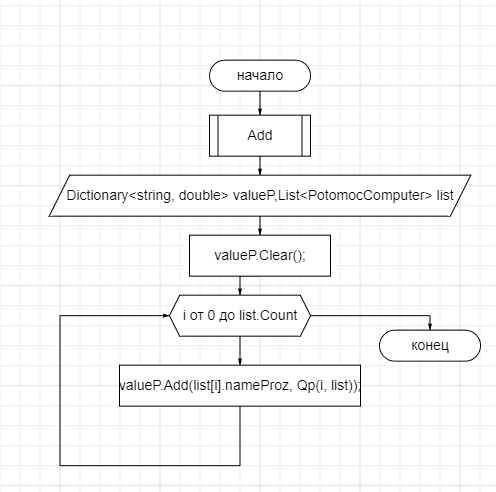


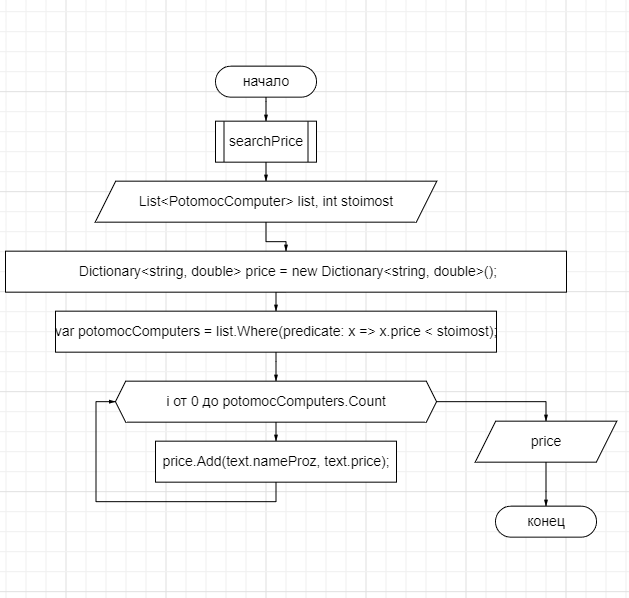
 

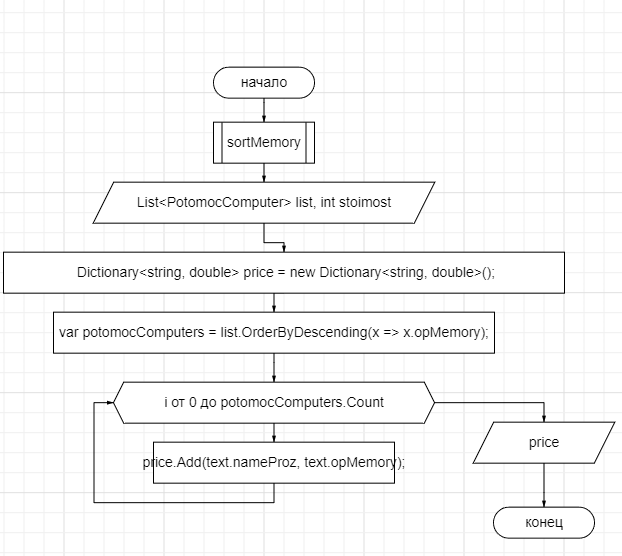












## **1.5 Используемые библиотеки**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

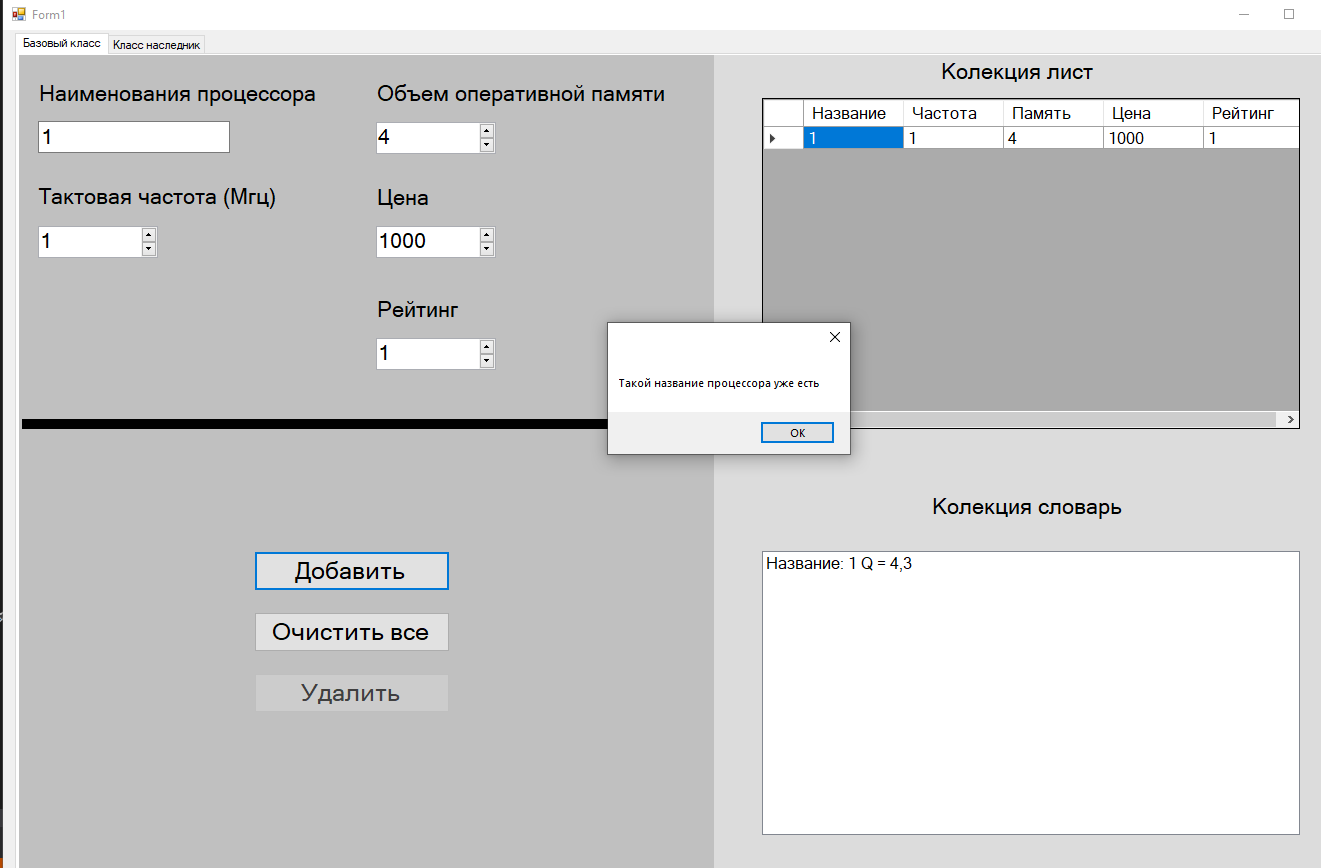
using System.Text;

using System.Threading.Tasks;

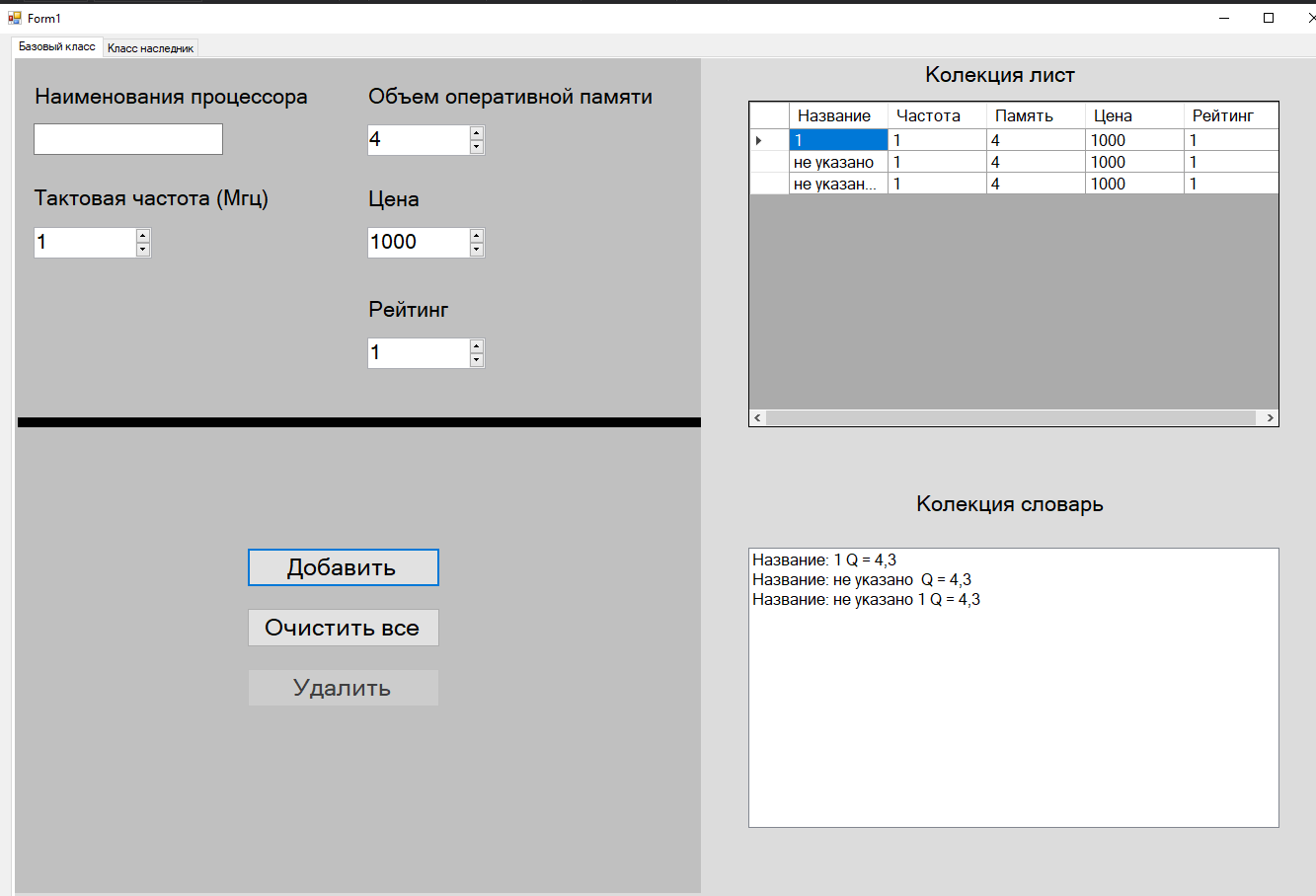
using System.Windows.Forms;

## **1.6 Тестовые случаи**

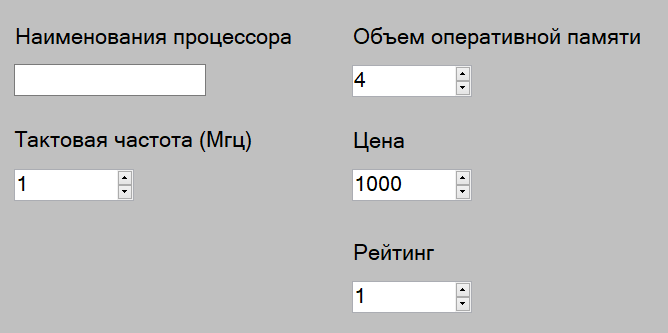
Пользователь не может добавить процессор с таким же именем

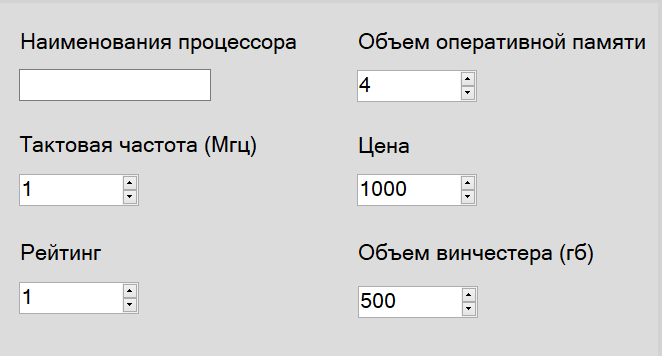


Если пользователь не ввел название процессора, то программа запишет имя, которое еще не было



Благодаря numericUoDown пользователь не может ввести значения, которые приведут к ошибке





## **1.7 Используемые инструменты**

Язык C#, Среда разработки Visual Studio, .NET Framework 4.7.2

## **1.8 Описание пользовательского интерфейса**

Textbox 2 шт – для ввода информации.

Listbox (4 шт) – для вывода словарей.

numericUpDown (9 шт) – для ввода информации

DataGridView – для вывода коллекий

naslAdd, addBut – кнопки для добавление информации в лист

clear, naslClear – кнопка для отчистки листа и словаря

naslDel, Delete – кнопка для удаление элемента из словаря и листа

## **1.9 Приложение (pr screen экранов)**

