Министерство образования и молодежной политики Свердловской области

ГАПОУ СО «Екатеринбургский колледж транспортного строительства»

Отчёт по учебной практике

УП 01.02

Выполнил: Рогов А.М.

Группа: ПР-21

Преподаватель: Мирошниченко Г.В.

2025

Содержание

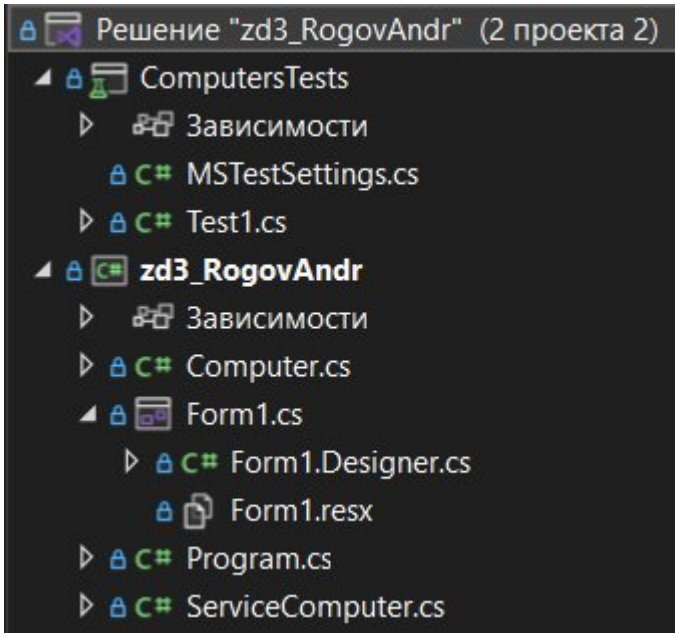
1. Задание №1 Приложение Windows Forms «Компьютеры»

1.1 Описание задачи

Создать проект для демонстрации работы: ввод-вывод информации об объектах базового класса «Компьютер» и класса потомка «Серверный компьютер». Используя методы Linq и коллекции. Также реализовать модульные тесты к реализованным методам класса.

1.2 Структура проекта

Структура проекта (рис. 1)



(рис. 1)

В решение входят:

- ComputersTest – модульные тесты к реализованным методам класса.
- zd3_RogovAndr – проект в котором находятся классы и форма.
- Form1.cs – код, который содержит в себе форма.
- Computer.cs – класс компьютера.
- ServiceComputer.cs – класс серверного компьютера.

1.3 Описание разработанных функций

- CalculateQuality() – считает качество компьютера.

```
public virtual double CalculateQuality()
{
    return 0.3 * _clockSpeed + _ramSize;
}
```

- GetInfo() – выводит информацию о компьютере.

```
public virtual string GetInfo()
{
    string graphics = _hasDiscreteGraphics ? "Имеется" : "Отсутствует";
    return $"Компьютер: {_processorName}, Частота: {_clockSpeed} МГц, ОЗУ: {_ramSize} МБ, " +
          $"Производитель: {_manufacturer}, Дискретная графика: {graphics}, Q: {CalculateQuality():F2}";
}
```

- AddComputer(Computer comp) – добавляет компьютер в список.

```
public void AddComputer(Computer comp)
{
    AllComputers.Add(comp);
    AddToDictionary(comp);
}
```

- AddComputer(Computer comp, int index) – добавляет компьютер по индексу.

```
public void AddComputer(Computer comp, int index)
{
    AllComputers.Insert(index, comp);
    AddToDictionary(comp);
}
```

- AddToDictionary(Computer comp) – добавляет в словарь

```
private void AddToDictionary(Computer comp)
{
    if (!ComputersByProcessor.ContainsKey(comp._processorName))
    {
        ComputersByProcessor[comp._processorName] = new List<Computer>();
    }
    ComputersByProcessor[comp._processorName].Add(comp);
}
```

- RemoveComputer() – удаляет последний компьютер.

```
public bool RemoveComputer()
{
    if (AllComputers.Count == 0) return false;

    var lastComp = AllComputers.Last();
    ComputersByProcessor[lastComp._processorName].Remove(lastComp);

    if (ComputersByProcessor[lastComp._processorName].Count == 0)
    {
        ComputersByProcessor.Remove(lastComp._processorName);
    }

    AllComputers.Remove(lastComp);
    return true;
}
```

- RemoveComputer(string processorName) – удаляет компьютеры по имени процессора.

```
public bool RemoveComputer(string processorName)
{
    if (string.IsNullOrEmpty(processorName) || !ComputersByProcessor.ContainsKey(processorName))
        return false;

    var computersToRemove = AllComputers
        .Where(c => c._processorName == processorName)
        .ToList();

    if (computersToRemove.Count == 0)
        return false;

    foreach (var comp in computersToRemove)
    {
        AllComputers.Remove(comp);
    }

    //Обновляем словарь
    ComputersByProcessor[processorName].Clear();
    if (ComputersByProcessor[processorName].Count == 0)
    {
        ComputersByProcessor.Remove(processorName);
    }

    return true;
}
```

- FindComputersByProcessor() – ищет компьютеры по процессору.

```
public List<Computer> FindComputersByProcessor(string processorName)
{
    if (ComputersByProcessor.TryGetValue(processorName, out var computers))
        return new List<Computer>(computers); // Возвращаем копию
    return null;
}
```

- CalculateQuality() – перегрузка в ServiceComputer, считает Qp с учетом HDD.

```
public override double CalculateQuality()
{
    return base.CalculateQuality() + 0.5 * _hddSize;
}
```

- GetInfo() – перегрузка в ServiceComputer, выводит информацию о серверном компьютере.

```
public override string GetInfo()
{
    string ssd = _hasSSD ? "Имеется" : "Отсутствует";
    return $"Серверный " + base.GetInfo() +
          $", HDD: {_hddSize} ГБ, Наличие SSD: {ssd}, Qp: {CalculateQuality():F2}";
}
```

- UpdateListBox() – обновляет листбокс.

```
private void UpdateListBox()
{
    numericUpDownIndex.Maximum = _computerManager.AllComputers.Count;
    listBoxComputers.Items.Clear();
    foreach (var comp in _computerManager.AllComputers)
    {
        listBoxComputers.Items.Add(comp.GetInfo());
    }
}
```

- ValidateInputs() – проверяет, что введены валидные данные.

```
private bool ValidateInputs()
{
    //Проверка полей
    if (string.IsNullOrWhiteSpace(textProcessorName.Text) ||
        string.IsNullOrWhiteSpace(textClockSpeed.Text) ||
        string.IsNullOrWhiteSpace(textRamSize.Text) ||
        string.IsNullOrWhiteSpace(textManufacturer.Text))
    {
        MessageBox.Show("Заполните все обязательные поля!", "Ошибка");
        return false;
    }

    //Проверка числовых значений
    if (!double.TryParse(textClockSpeed.Text, out double speed) || speed <= 0 ||
        !int.TryParse(textRamSize.Text, out int ram) || ram <= 0)
    {
        MessageBox.Show("Частота и ОЗУ должны быть положительными числами!", "Ошибка");
        return false;
    }

    //Для серверного компьютера проверяем HDD
    if (rbService.Checked && (!int.TryParse(textHddSize.Text, out int hdd) || hdd <= 0))
    {
        MessageBox.Show("Объем HDD должен быть положительным числом!", "Ошибка");
        return false;
    }

    return true;
}
```

- btnAdd_Click() – кнопка добавления обычного компа.

```
private void btnAdd_Click(object sender, EventArgs e)
{
    if (!ValidateInputs()) return;

    if (rbBasic.Checked)
    {
        var computer = new Computer(
            textProcessorName.Text,
            double.Parse(textClockSpeed.Text),
            int.Parse(textRamSize.Text),
            textManufacturer.Text,
            chkDiscrete.Checked);

        _computerManager.AddComputer(computer);
    }
    else
    {
        var serviceComp = new ServiceComputer(
            textProcessorName.Text,
            double.Parse(textClockSpeed.Text),
            int.Parse(textRamSize.Text),
            textManufacturer.Text,
            chkDiscrete.Checked,
            int.Parse(textHddSize.Text),
            chkSSD.Checked);

        _computerManager.AddComputer(serviceComp);
    }

    UpdateListBox();
}
```

- btnAddAtIndex_Click() – кнопка добавления по индексу.

```
private void btnAddAtIndex_Click(object sender, EventArgs e)
{
    if (!ValidateInputs()) return;

    int index = (int)numericUpDownIndex.Value;

    if (rbBasic.Checked)
    {
        var computer = new Computer(
            textProcessorName.Text,
            double.Parse(textClockSpeed.Text),
            int.Parse(textRamSize.Text),
            textManufacturer.Text,
            chkDiscrete.Checked);

        _computerManager.AddComputer(computer, index);
    }
    else
    {
        var serviceComp = new ServiceComputer(
            textProcessorName.Text,
            double.Parse(textClockSpeed.Text),
            int.Parse(textRamSize.Text),
            textManufacturer.Text,
            chkDiscrete.Checked,
            int.Parse(textHddSize.Text),
            chkSSD.Checked);

        _computerManager.AddComputer(serviceComp, index);
    }

    UpdateListBox();
}
```

- btnDeleteLast_Click() – удаляет последний комп.

```
private void btnDeleteLast_Click(object sender, EventArgs e)
{
    if (_computerManager.RemoveComputer())
    {
        UpdateListBox();
    }
    else
    {
        MessageBox.Show("Нет компьютеров для удаления", "Информация");
    }
}
```

- btnDeleteName_Click() – удаляет по имени процессора.

```
private void btnDeleteName_Click(object sender, EventArgs e)
{
    if (string.IsNullOrWhiteSpace(textProcessorName.Text))
    {
        MessageBox.Show("Введите название процессора", "Ошибка");
        return;
    }

    if (_computerManager.RemoveComputer(textProcessorName.Text))
    {
        UpdateListBox();
    }
    else
    {
        MessageBox.Show("Компьютер с таким процессором не найден", "Ошибка");
    }
}
```

- btnSearchByProcessor_Click() – ищет компьютеры по процессору.

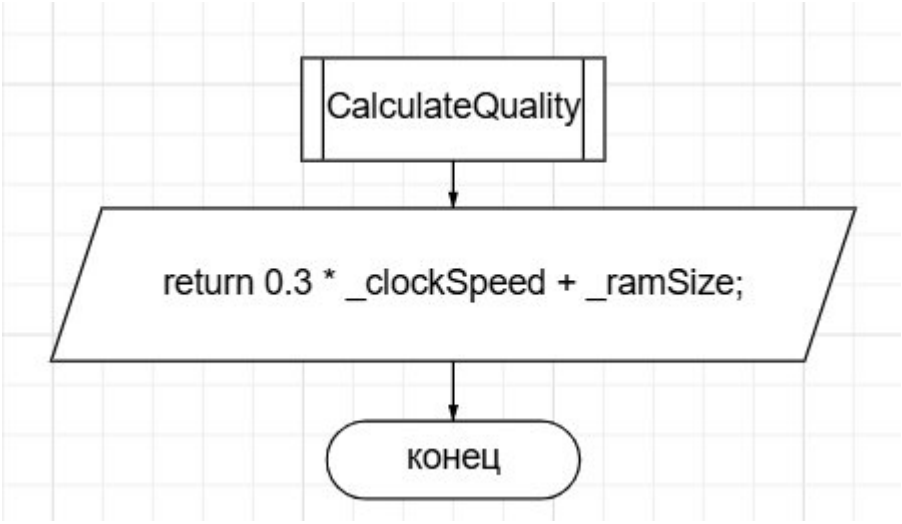
```
private void btnSearchByProcessor_Click(object sender, EventArgs e)
{
    string searchName = textSearchProcessor.Text;

    if (string.IsNullOrWhiteSpace(searchName))
    {
        MessageBox.Show("Введите название процессора для поиска");
        return;
    }

    listBoxComputers.Items.Clear();

    var computers = _computerManager.FindComputersByProcessor(searchName);
    if (computers != null)
    {
        foreach (var comp in computers)
        {
            listBoxComputers.Items.Add(comp.GetInfo());
        }
    }
    else
    {
        MessageBox.Show($"Компьютеров с процессором {searchName} не найдено");
    }
}
```

- rbBasic_CheckedChanged() – меняет доступность полей при переключении радио кнопок.

```
private void rbBasic_CheckedChanged(object sender, EventArgs e)
{
    textHddSize.Enabled = !rbBasic.Checked;
    chkSSD.Enabled = !rbBasic.Checked;
}
```

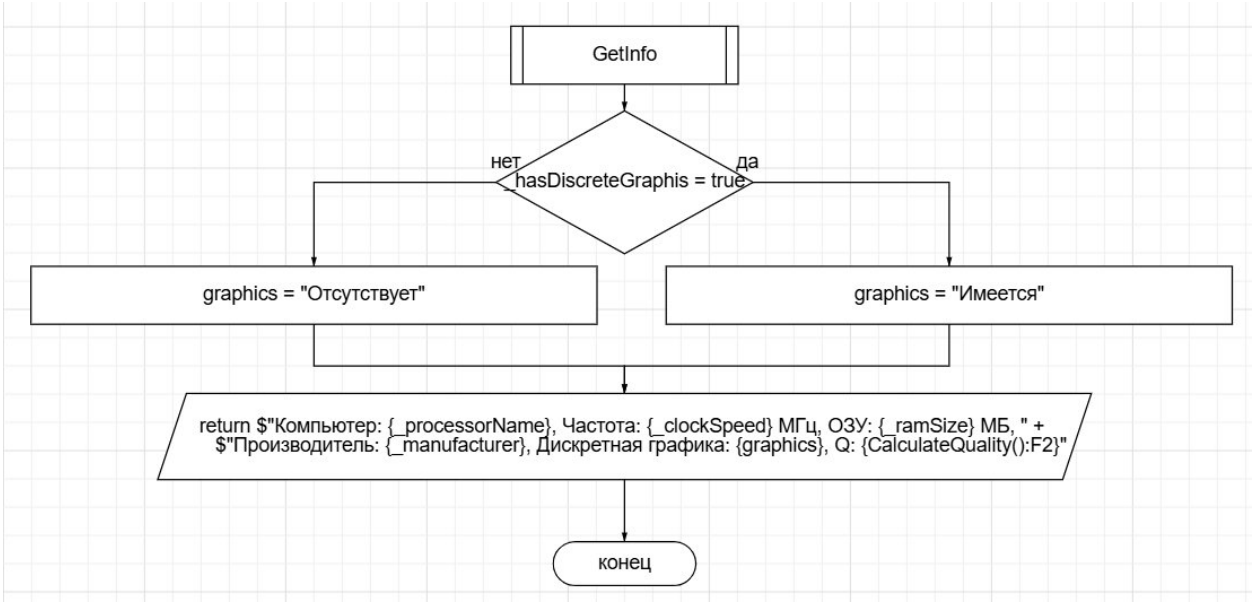
1.4Алгоритм решения

Метод CalculateQuality() (рис. 2)



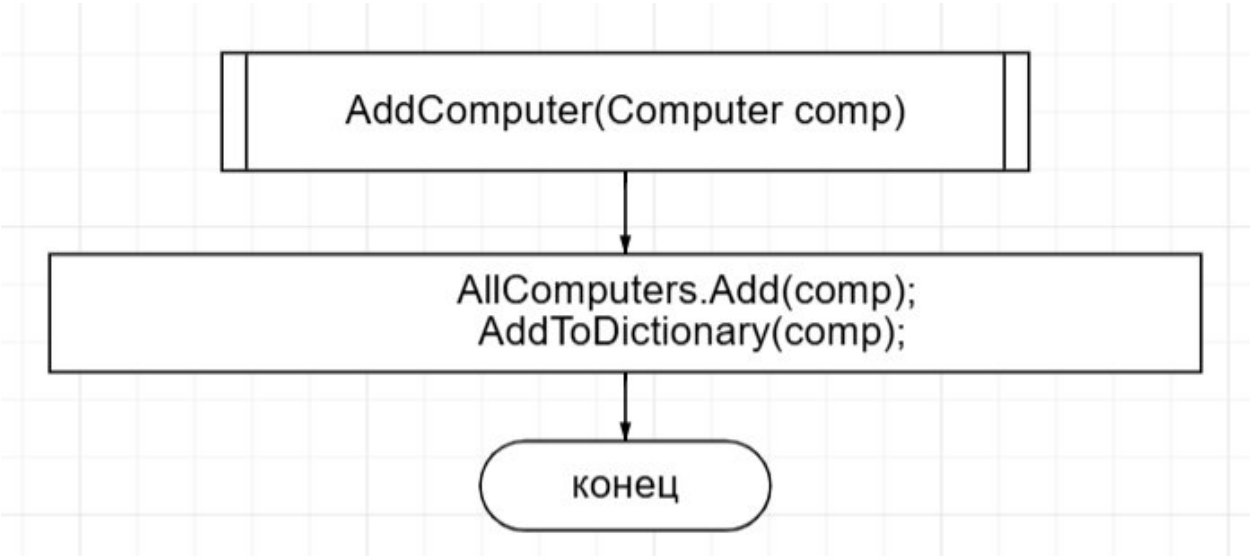
(рис. 2)

Метод GetInfo() (рис. 3)



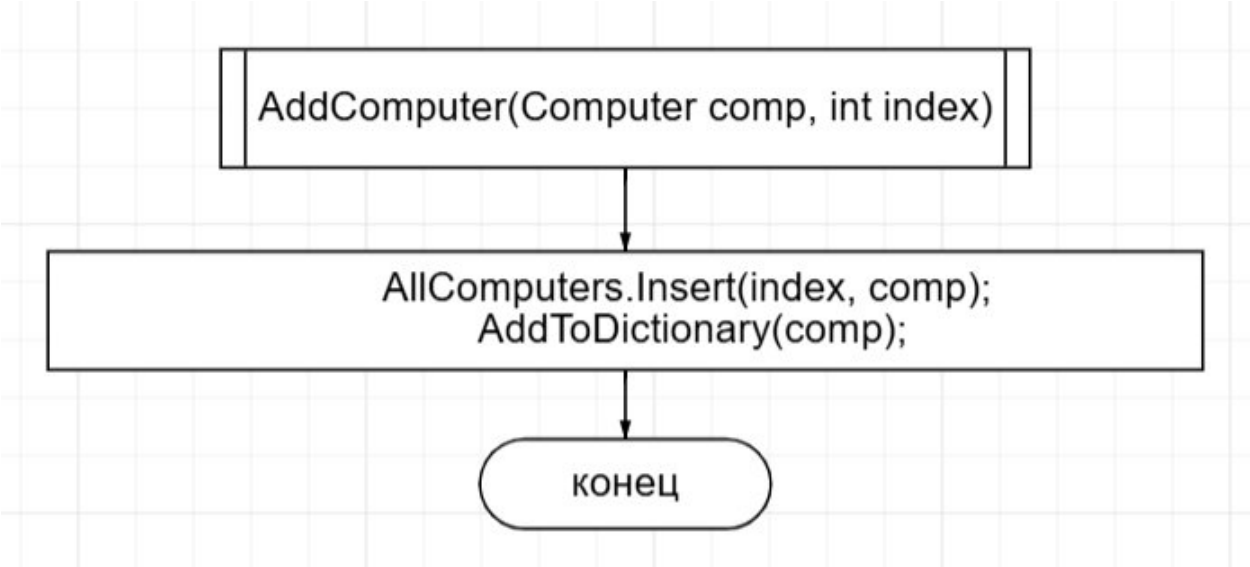
(рис. 3)

Метод AddComputer(Computer comp) (рис. 4)



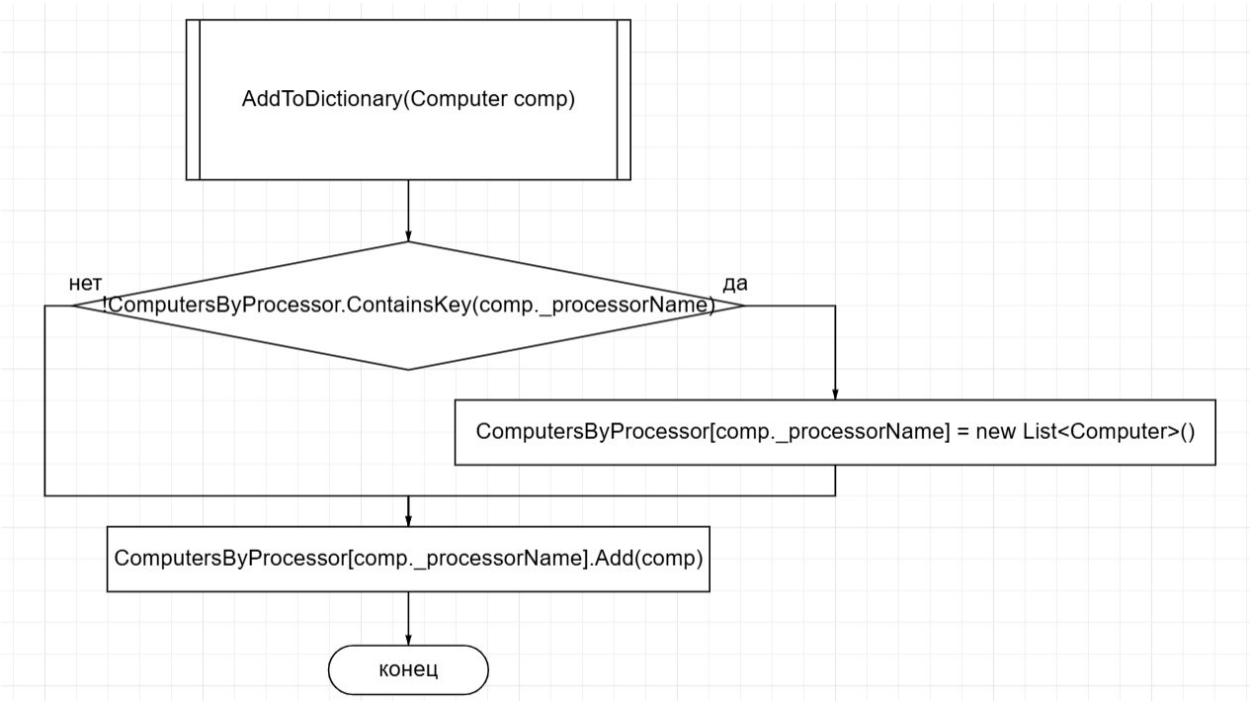
(рис. 4)

Перегрузка для метода AddComputer(Computer comp, int index) (рис. 5)



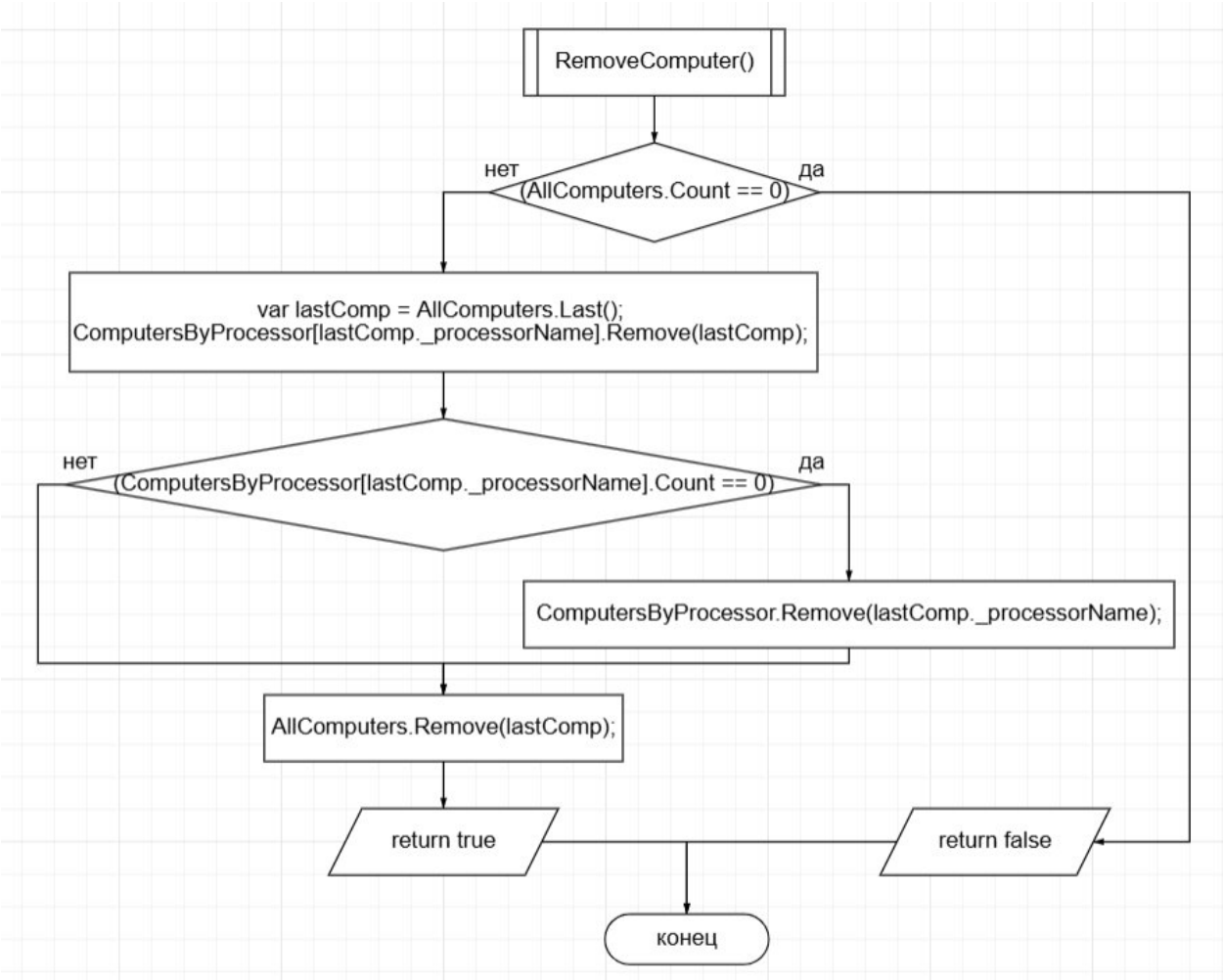
(рис. 5)

Метод AddToDictionary(Computer comp) (рис. 6)



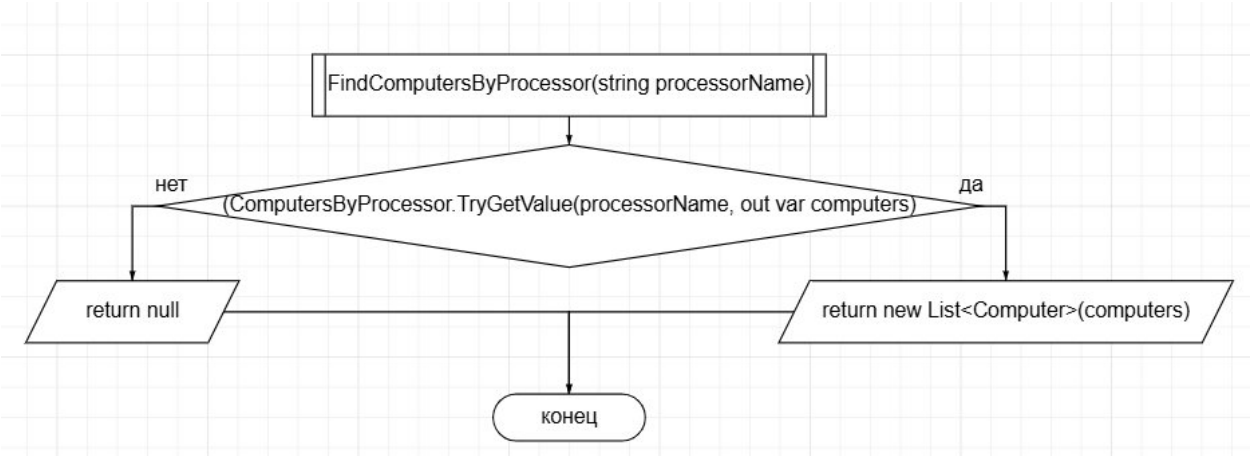
(рис. 6)

Метод RemoveComputer() (рис. 7)



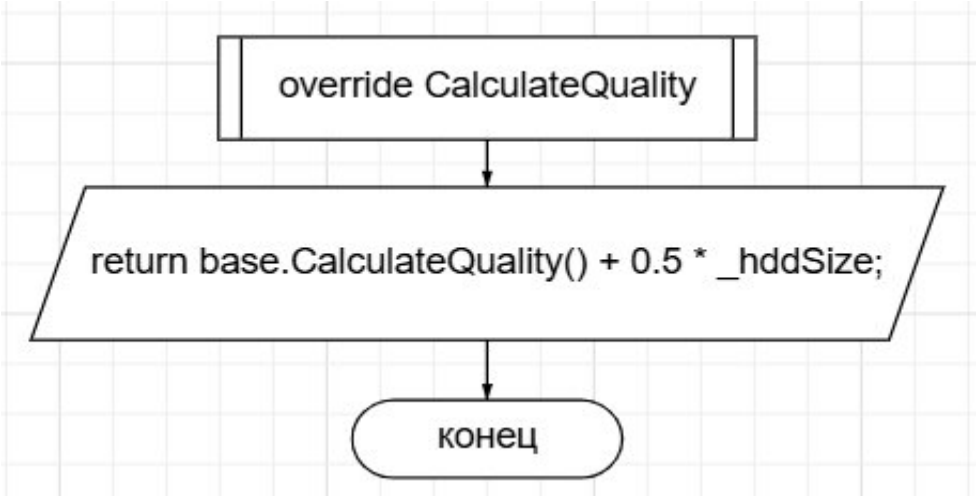
(рис. 7)

Метод FindComputersByProcessor(string processorName) (рис. 8)



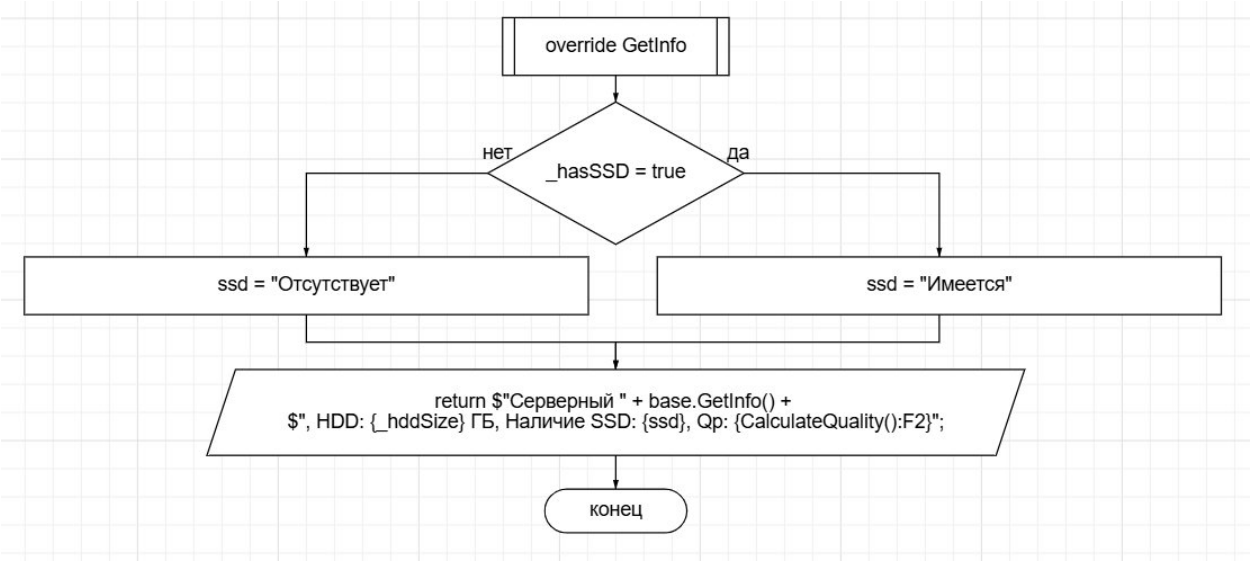
(рис. 8)

Перегрузка метода CalculateQuality() в классе-потомке (рис. 9)



(рис. 9)

Перегрузка метода GetInfo() в классе-потомке (рис. 10)



(рис. 10)

1.5 Используемые библиотеки

В проекте использовались библиотеки:

- Collections.Generic;
- Linq;
- Text;
- Threading.Tasks;

1.6 Тестовые случаи

Проверка на добавление компьютера, ожидается появление одного компьютера (рис. 11)

```
public void ДобавлениеКомпьютера_КорректныеДанные_УспешноеДобавление()
{
    var computer = new Computer("Intel i7", 3.5, 16, "Dell", true);

    _computerManager.AddComputer(computer);

    Assert.AreEqual(1, _computerManager.AllComputers.Count,
        $"Ожидался 1 компьютер, но найдено {_computerManager.AllComputers.Count}.");
    Assert.AreEqual("Intel i7", _computerManager.AllComputers[0].GetProcessorName());
}
```

(рис. 11)

Проверка на добавление компьютера по индексу, ожидается, что добавление по индексу добавит в начало списка (рис. 12)

```
public void ДобавлениеКомпьютераПоИндексу_ВалидныйИндекс_КорректнаяВставка()
{
    var comp1 = new Computer("CPU1", 2.5, 8, "Brand1", false);
    var comp2 = new Computer("CPU2", 3.0, 16, "Brand2", true);
    _computerManager.AddComputer(comp1);
    _computerManager.AddComputer(comp2, 0);

    Assert.AreEqual(2, _computerManager.AllComputers.Count);
    Assert.AreEqual("CPU2", _computerManager.AllComputers[0].GetProcessorName());
    Assert.AreEqual("CPU1", _computerManager.AllComputers[1].GetProcessorName());
    Assert.IsTrue(_computerManager.ComputersByProcessor.ContainsKey("CPU2"));
}
```

(рис. 12)

Проверка на добавление по индексу, ожидается добавление в конец списка (рис. 13)

```
public void ДобавлениеКомпьютераПоИндексу_ИндексРавенКоличеству_ДобавлениеВКонец()
{
    var comp1 = new Computer("CPU1", 2.5, 8, "Brand1", false);
    var comp2 = new Computer("CPU2", 3.0, 16, "Brand2", true);
    _computerManager.AddComputer(comp1);

    _computerManager.AddComputer(comp2, 1);

    Assert.AreEqual(2, _computerManager.AllComputers.Count);
    Assert.AreEqual("CPU2", _computerManager.AllComputers[1].GetProcessorName());
}
```

(рис. 13)

Проверка на добавление в словарь, ожидается добавление в словарь одного компьютера (рис. 14)

```
public void ДобавлениеВСловарь_НовыйПроцессор_СозданиеНовойЗаписи()
{
    var comp = new Computer("NewCPU", 2.5, 8, "Brand", false);

    _computerManager.AddComputer(comp);

    Assert.IsTrue(_computerManager.ComputersByProcessor.ContainsKey("NewCPU"));
    Assert.AreEqual(1, _computerManager.ComputersByProcessor["NewCPU"].Count);
}
```

(рис. 14)

Проверка на добавление в словарь, ожидается увеличение количества объектов до двух (рис. 15)

```
public void ДобавлениеВСловарь_СуществующийПроцессор_ДобавлениеВСуществующийСписок()
{

    var comp1 = new Computer("SameCPU", 2.5, 8, "Brand1", false);
    var comp2 = new Computer("SameCPU", 3.0, 16, "Brand2", true);
    _computerManager.AddComputer(comp1);

    _computerManager.AddComputer(comp2);

    Assert.AreEqual(1, _computerManager.ComputersByProcessor.Count);
    Assert.AreEqual(2, _computerManager.ComputersByProcessor["SameCPU"].Count);
}
```

(рис. 15)

Проверка на удаление компьютера, ожидается True, если список пусто (рис. 16)

```
public void УдалениеКомпьютера_СуществующийКомпьютер_ВозвращаетTrue()
{
    var computer = new Computer("Intel i7", 3.5, 16, "Dell", true);
    _computerManager.AddComputer(computer);

    var result = _computerManager.RemoveComputer();

    Assert.IsTrue(result);
    Assert.AreEqual(0, _computerManager.AllComputers.Count);
}
```

(рис. 16)

Проверка на удаление компьютера, ожидается false, если список пустой (рис. 17)

```
public void УдалениеКомпьютера_ПустойСписок_ВозвращаетFalse()
{
    var result = _computerManager.RemoveComputer();

    Assert.IsFalse(result);
}
```

(рис. 17)

Проверка на удаление компьютера по имени процессора, ожидается true и пустой список (рис. 18)

```
public void УдалениеКомпьютера_ПоИмениПроцессора_УспешноеУдаление()
{
    var manager = new Computer("", 0, 0, "", false);
    manager.AddComputer(new Computer("Intel i5", 2.5, 8, "Dell", false));

    bool result = manager.RemoveComputer("Intel i5");


    Assert.IsTrue(result);
    Assert.AreEqual(0, manager.AllComputers.Count);
}
```

(рис. 18)

Проверка на удаление компьютера по имени процессора, ожидается false, если не существует компьютера (рис. 19)

```
public void УдалениеКомпьютера_НесуществующийПроцессор_ВозвращаетFalse()
{
    var manager = new Computer("", 0, 0, "", false);

    bool result = manager.RemoveComputer("AMD Ryzen");

    Assert.IsFalse(result);
}
```

(рис. 19)

Проверка поиска компьютера по процессору, ожидается список из двух компьютеров с процессором Ryzen 5 (рис. 20)

```
public void ПоискКомпьютеровПоПроцессору_СуществующийПроцессор_ВозвращаетСписок()
{
    var computer1 = new Computer("Ryzen 5", 3.2, 8, "HP", false);
    var computer2 = new Computer("Ryzen 5", 3.4, 16, "Lenovo", true);
    _computerManager.AddComputer(computer1);
    _computerManager.AddComputer(computer2);

    var result = _computerManager.FindComputersByProcessor("Ryzen 5");

    Assert.AreEqual(2, result.Count);
}
```

(рис. 20)

Проверка расчёта качества обычного компьютера, ожидается значение 8.75 (рис. 21)

```
public void РасчетКачества_ОбычныйКомпьютер_ВозвращаетКорректноеЗначение()
{
    var computer = new Computer("Intel i5", 2.5, 8, "Acer", false);

    var quality = computer.CalculateQuality();

    Assert.AreEqual(8.75, quality);
}
```

(рис. 21)

Проверка расчёта качества серверного компьютера, ожидается значение 1032.9 (рис. 22)

```
public void РасчетКачества_СерверныйКомпьютер_ВозвращаетКорректноеЗначение()
{
    var serviceComputer = new ServiceComputer("Xeon", 3.0, 32, "IBM", true, 2000, true);

    var quality = serviceComputer.CalculateQuality();

    Assert.AreEqual(1032.9, quality, 0.001);
}
```

(рис. 22)

Проверка на получение информации о серверном компьютере, ожидается три определённые строки (рис. 23)

```
public void ПолучениеИнформации_СерверныйКомпьютер_ВозвращаетКорректнуюСтроку()
{
    var serviceComputer = new ServiceComputer("Xeon", 3.0, 32, "IBM", true, 2000, true);

    var info = serviceComputer.GetInfo();

    StringAssert.Contains(info, "Серверный Компьютер: Xeon");
    StringAssert.Contains(info, "HDD: 2000 ГБ");
    StringAssert.Contains(info, "Наличие SSD: Имеется");
}
```

(рис. 23)

Проверка на добавление серверного компьютера в список, ожидается добавление одного компьютера, и его тип равный ServiceComputer (рис. 24)

```
public void ДобавлениеСерверногоКомпьютера_ВОбщийСписок_РаботаетКорректно()
{
    var serviceComputer = new ServiceComputer("Xeon", 3.0, 32, "IBM", true, 2000, true);

    _computerManager.AddComputer(serviceComputer);

    Assert.AreEqual(1, _computerManager.AllComputers.Count);
    Assert.IsTrue(_computerManager.AllComputers[0] is ServiceComputer);
}
```

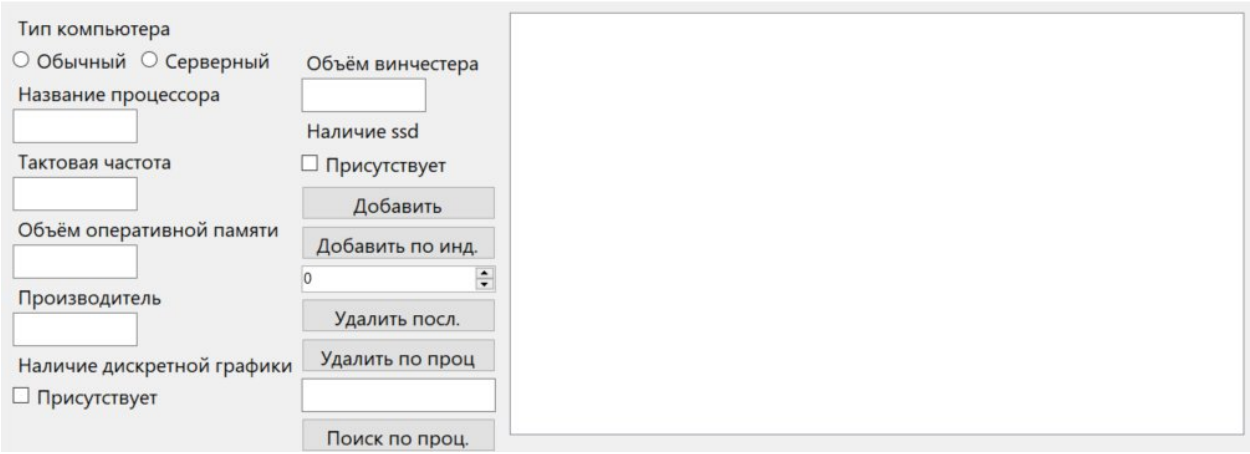
(рис. 24)

1.7 Используемые инструменты

Приложение Windows Forms .Net Framework, C#.

1.8 Описание пользовательского интерфейса

Вид формы (рис. 25)



(рис. 25)

Listbox показывает информацию об объектах.

Выбор типа компьютера отвечает за то, какой объект будет создан и передан в listbox.

Кнопка «добавить» добавляет введенные элементы по порядку.

Кнопка «добавить по индексу» смотрит какой выбран индекс в numericUpDown и добавляет по этому индексу в listbox.

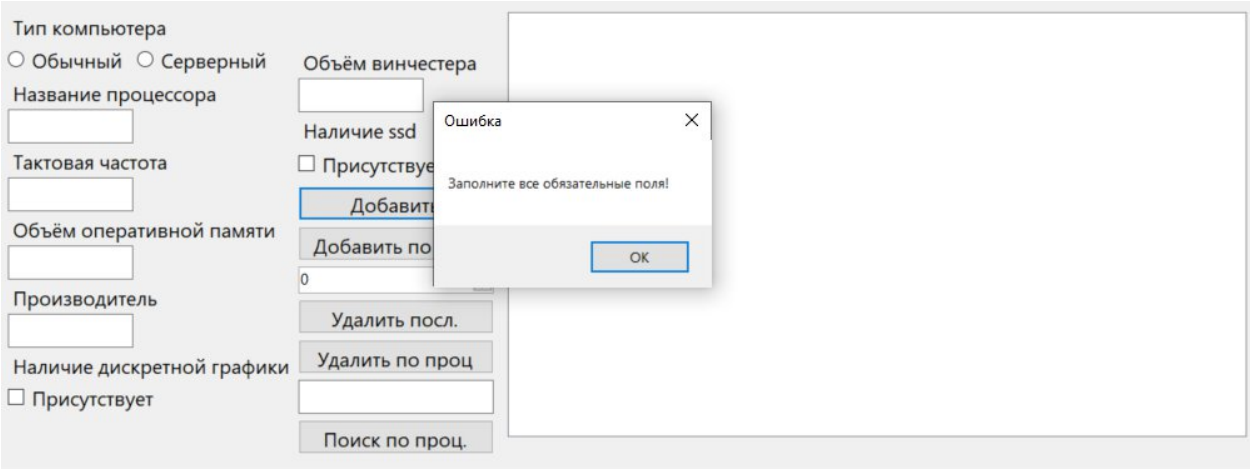
Кнопка «удалить последний» удаляет последний элемент.

Кнопка «удалить по процессору» удаляет компьютеры с выбранным процессором.

Кнопка «поиск по процессору» показывает в listbox компьютеры с подходящим процессором.

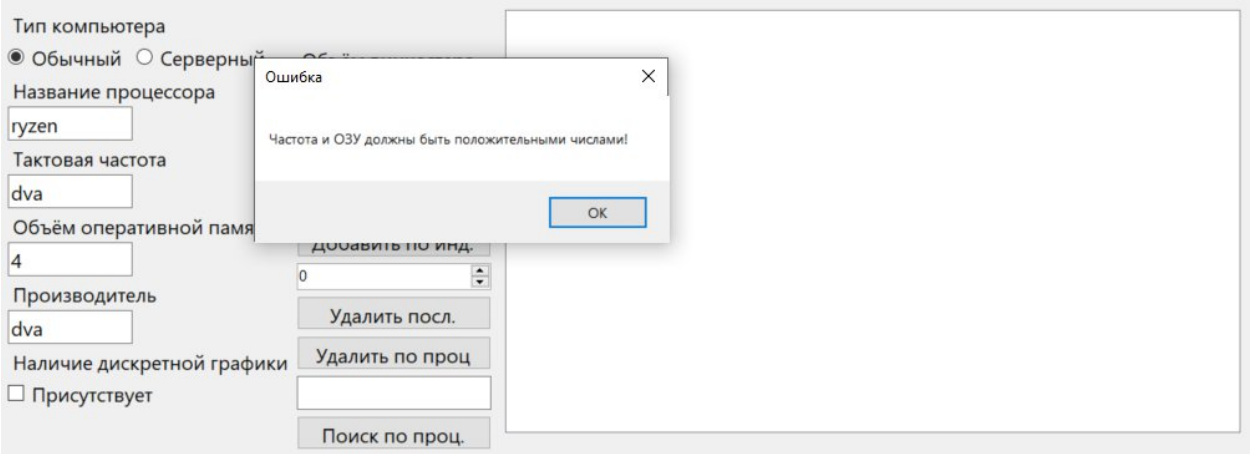
1.9 Приложение (pr screen экранов)

Ошибка при пустых полях (рис. 26).



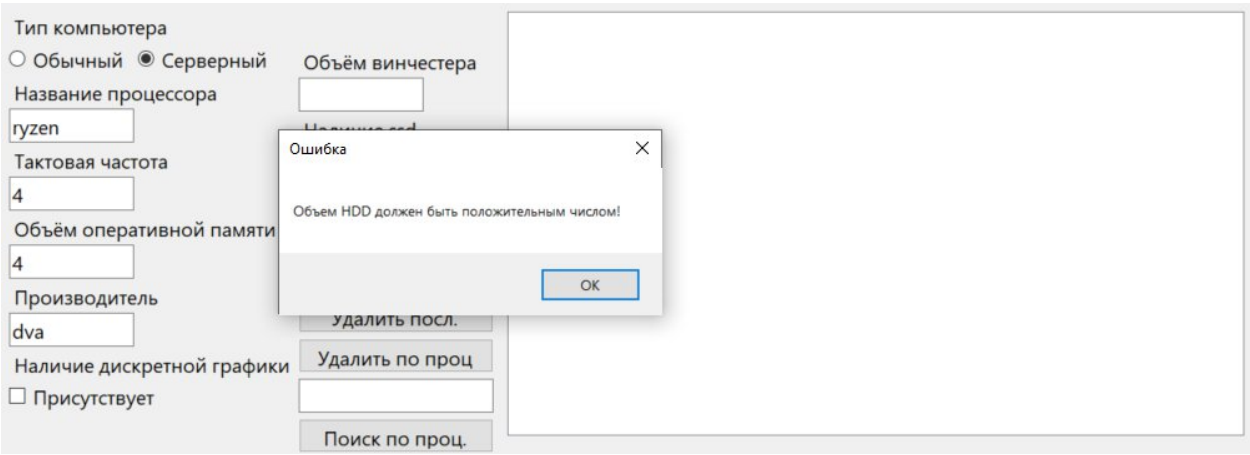
(рис. 26)

Ошибка при не числовом типе в чистоте и ОЗУ (рис. 27).



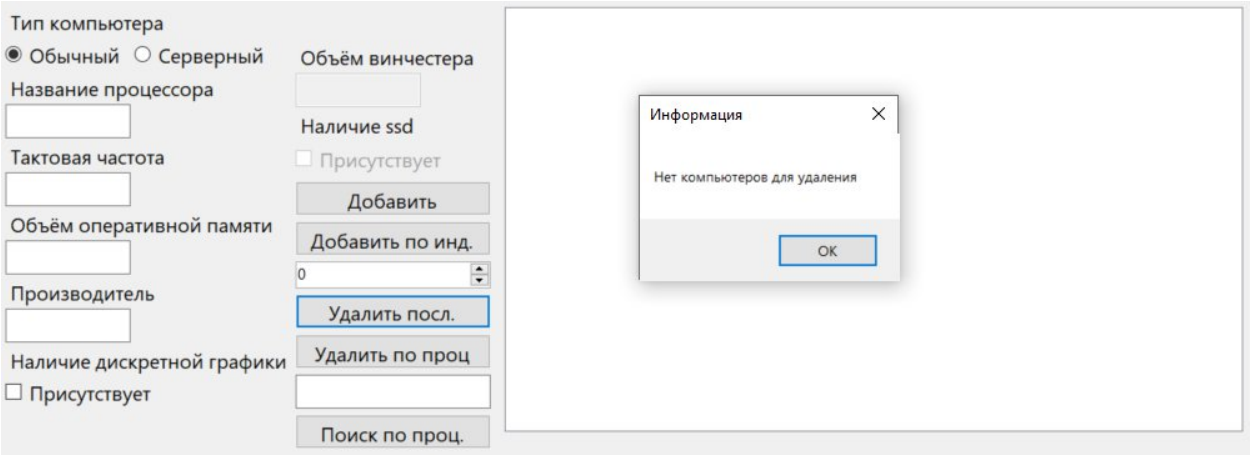
(рис. 27)

Ошибка при добавлении серверного компьютера без указания объёма HDD (рис. 28)



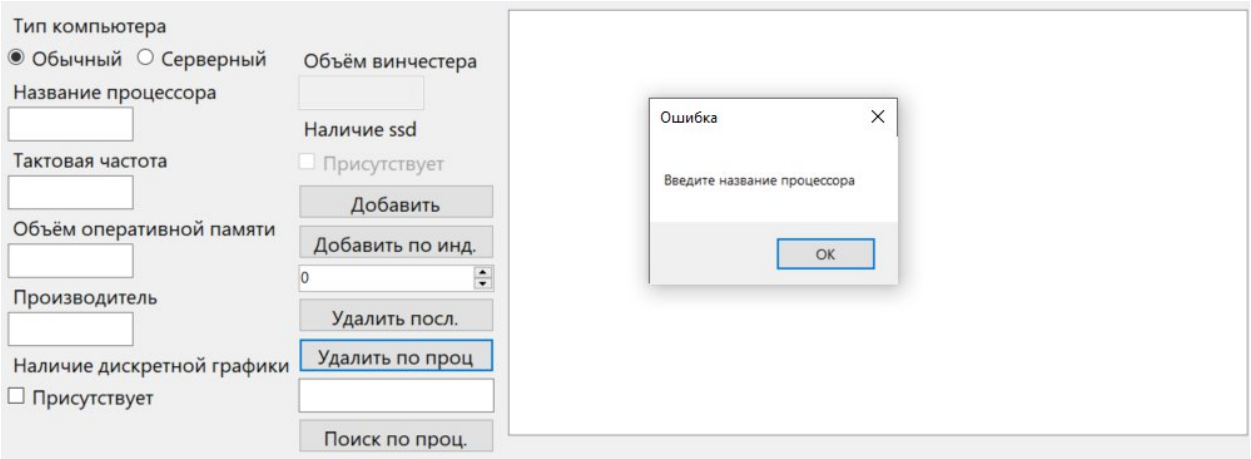
(рис. 28)

Ошибка при удалении последнего в пустом списке (рис. 29)



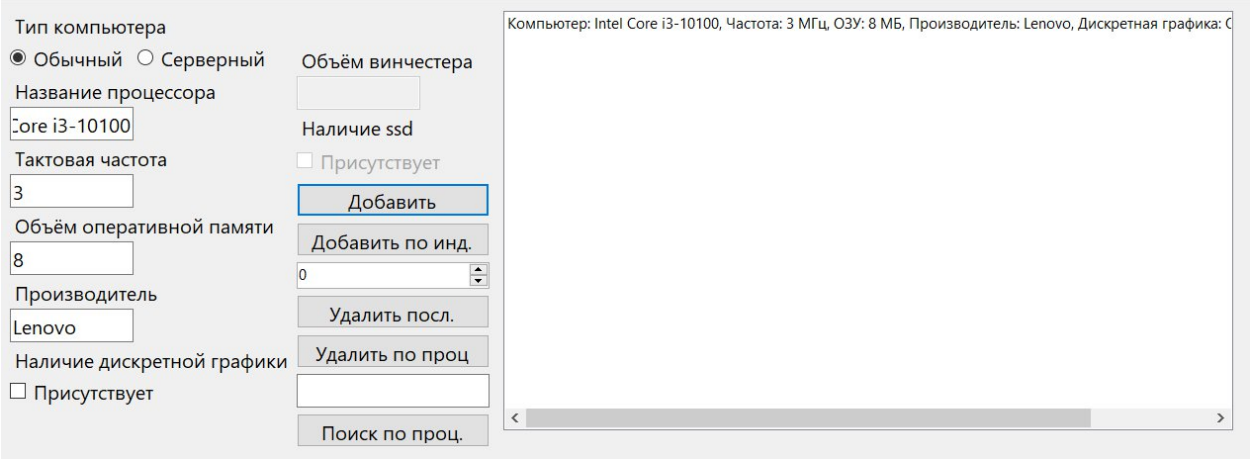
(рис. 29)

Ошибка при удалении по названию процессора при пустом поле ввода (рис. 30)



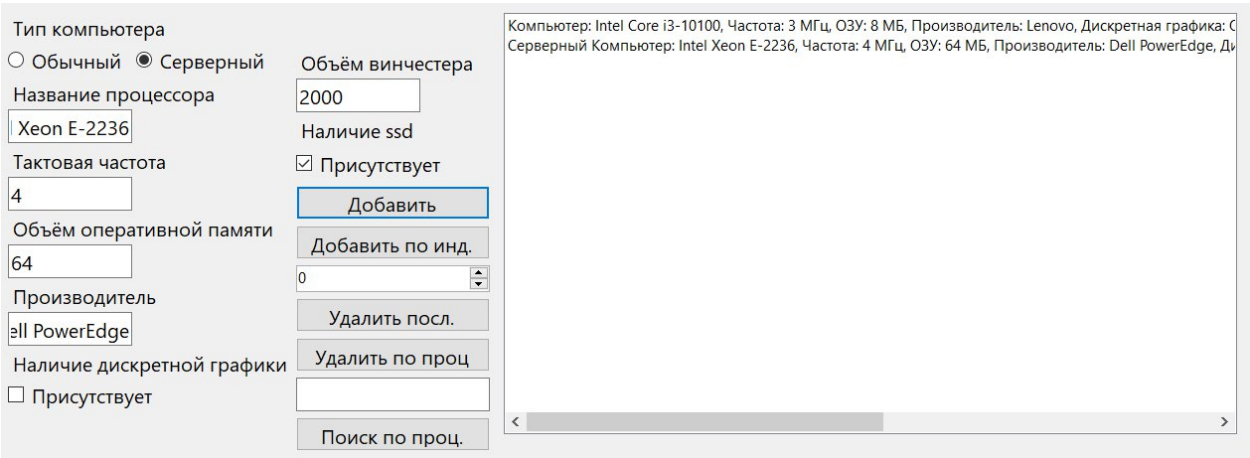
(рис. 30)

Добавление объекта обычного компьютера (рис. 31)



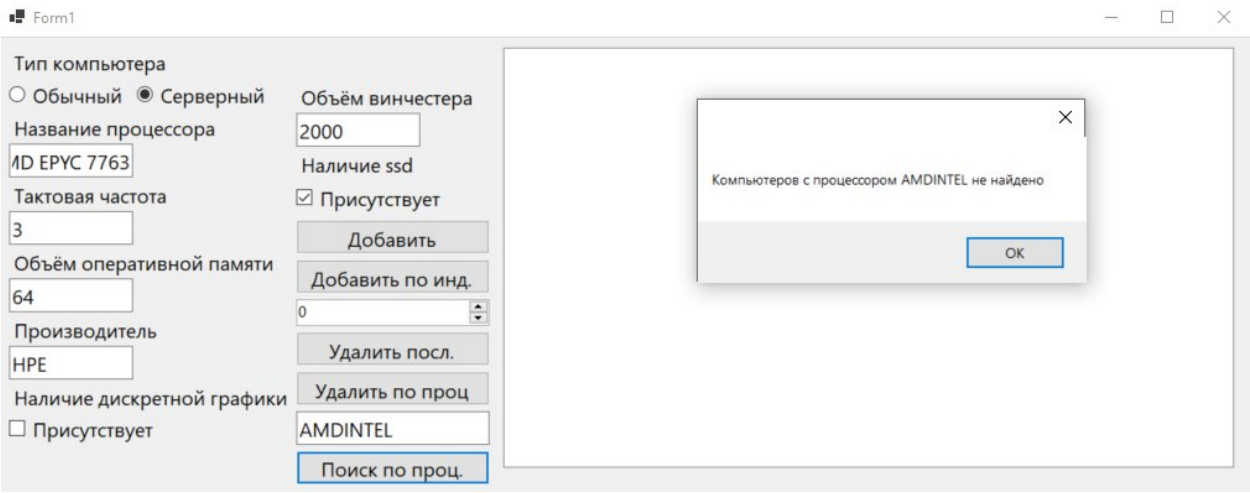
(рис. 31)

Добавление объекта серверного компьютера (рис. 32)



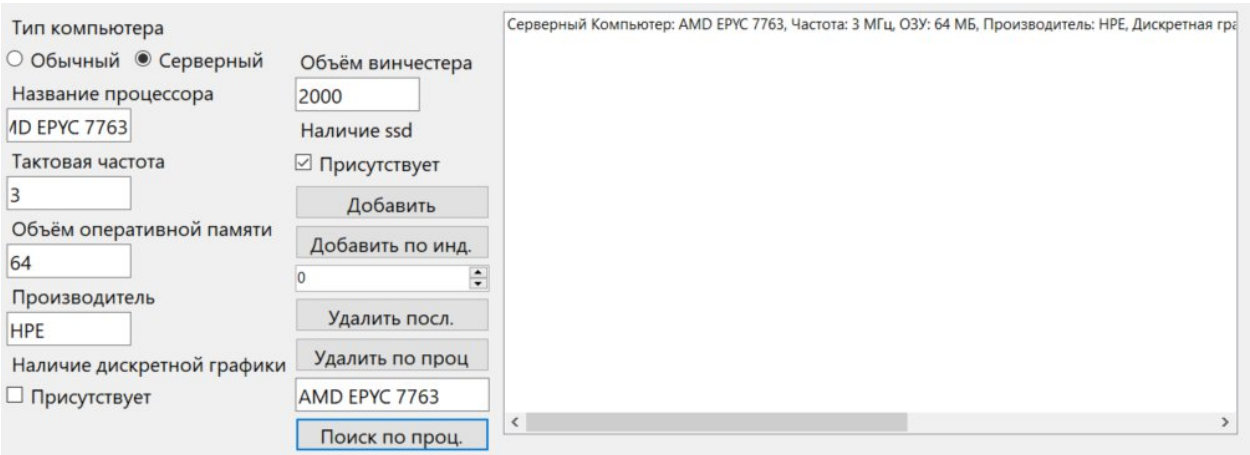
(рис. 32)

Ошибка при отсутствии процессора который ищем(рис. 33)



(рис. 33)

Поиск компьютеров по конкретному процессору (рис. 34)



(рис. 34)