Министерство транспорта Российской Федерации

Федеральное государственное автономное образовательное

учреждение высшего образования

«Российский университет транспорта»

(ФГАОУ ВО РУТ(МИИТ), РУТ (МИИТ)

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

Лабораторная работа № 19

по дисциплине: «Программирование и основы алгоритмизации»

на тему: «Запросы LINQ»

Выполнил: ст. гр. ТУУ-211

Кругликов Е.А.

Вариант №8

Проверил: к.т.н., доц. Сафронов А.И.

1. **Формулировка цели работы**

Освоить навыки подготовки наборов данных по нечётко сформулированному описанию предметной области при учёте особенностей запросов, отклики на которые необходимо предоставлять заявленным (потенциальным) пользователям программного обеспечения.

1. **Описание задачи**

Описание предметной области для подготовки данных. Пользователями являются сотрудники гостиницы и клиенты-постояльцы. В гостинице имеются номера, которые различаются по количеству мест, количеству комнат, удобствам и т.д. От всех этих параметров и их сочетания зависит цена гостиничного номера. Посетители могут занимать номера, бронировать их заранее. Постоянным посетителям и детям до 12 лет с родителями предоставляется скидка.

Реализовать запросы LINQ:

– предоставлять (и бронировать) посетителям номер в гостинице в зависимости от их потребностей (количества мест для размещения группы гостей и т.д.);

– показывать количество свободных (занятых) мест по категории номеров указанного типа;

– показывать список постоянных посетителей и предоставляемую им скидку;

– показывать информацию по указанному номеру (количество мест, комнат, наличие холодильника, телевизора, бара и т.д.).

1. **Сеть Петри запрограммированного технологического процесса**



Рисунок 1 – Сеть Петри запрограммированного технологического процесса, часть 1



Рисунок 2 – Сеть Петри запрограммированного технологического процесса, часть 2



Рисунок 3 – Сеть Петри запрограммированного технологического процесса, часть 3



Рисунок 4 – Сеть Петри запрограммированного технологического процесса, часть 4

1. **Схемы алгоритмов методов в составе решения, отмеченных на сети Петри в качестве «эффектов» (метка )**

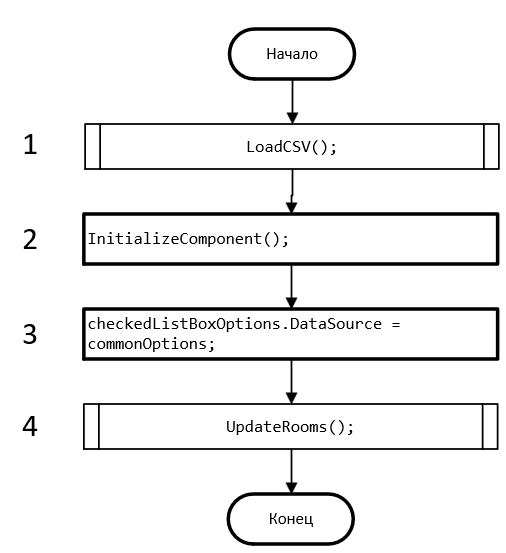


Рисунок 5 – Инициализация программы и загрузка данных из csv-файла

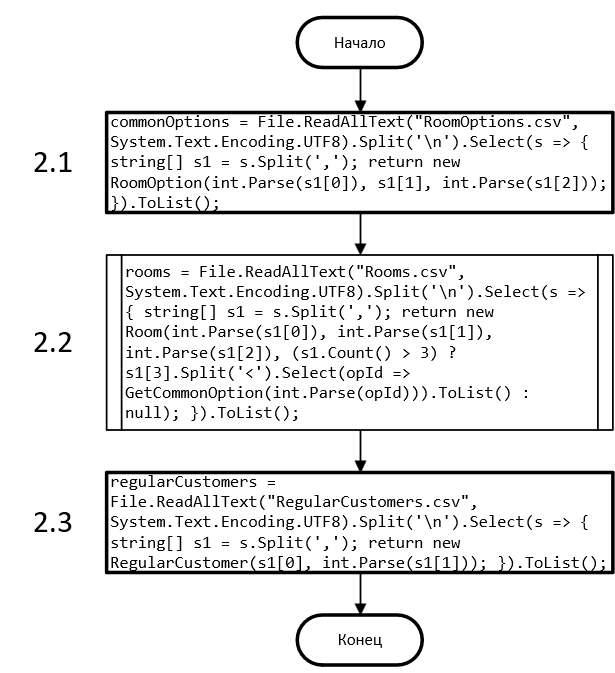


Рисунок 6 – Метод LoadCSV

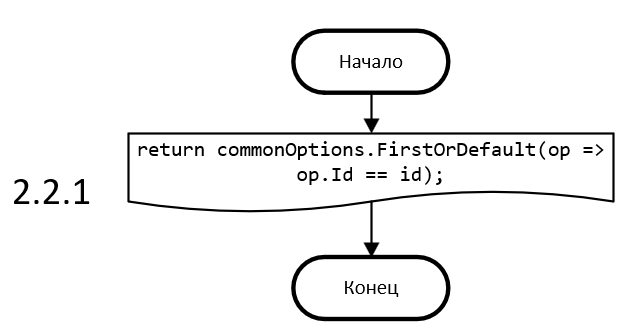


Рисунок 7 – Метод GetCommonOption

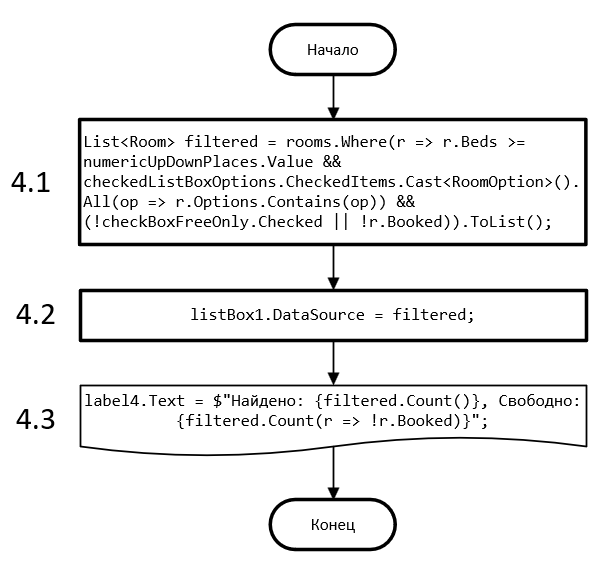


Рисунок 8 – Обновление списка найденных комнат в соответствии с фильтрами

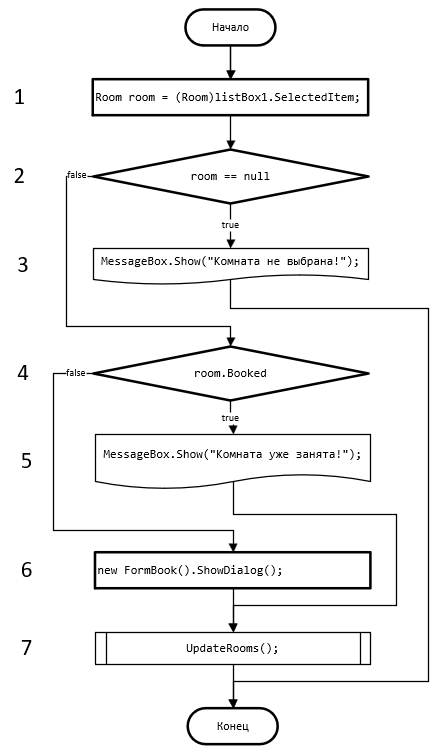


Рисунок 9 – Попытка открыть форму бронирования

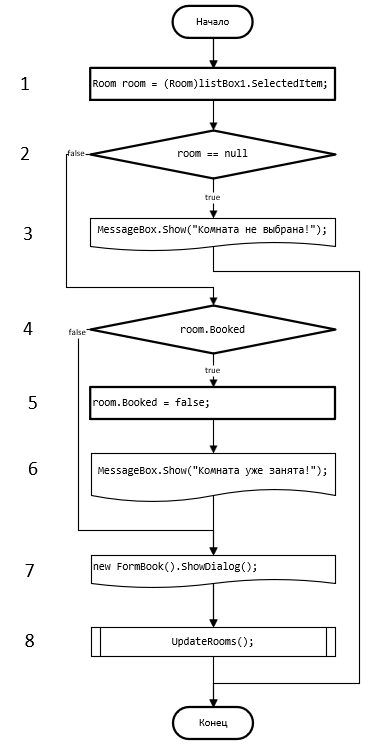


Рисунок 10 – Попытка освободить комнату

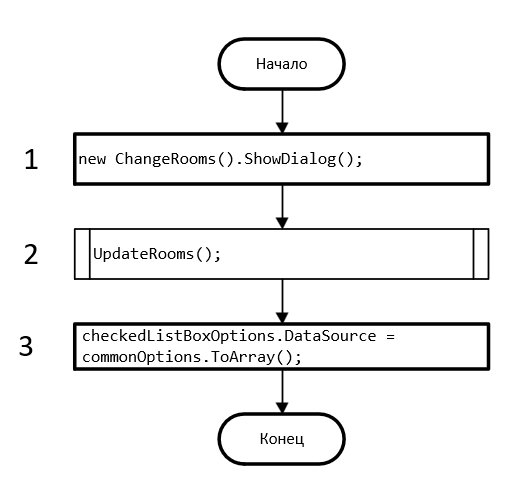


Рисунок 11 – Нажата кнопка «Изменить комнаты»

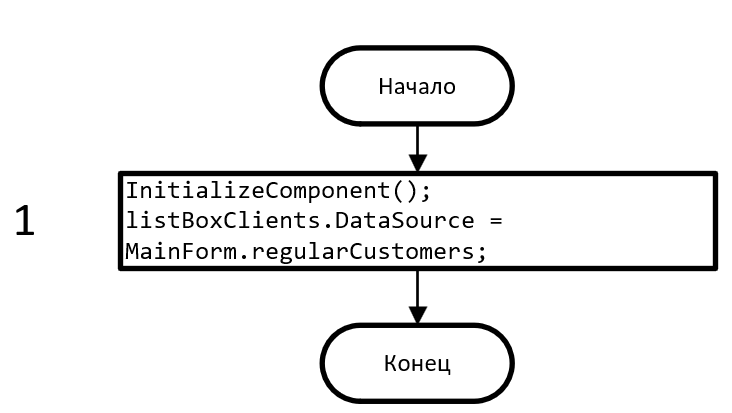


Рисунок 12 – Обновление списка постоянных покупателей

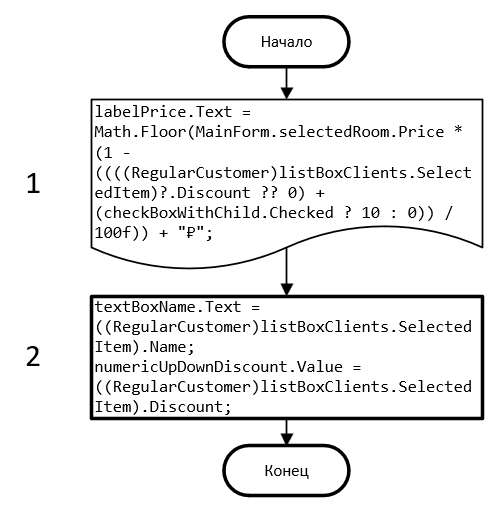


Рисунок 13 – Обновление полей покупателя и суммы к оплате

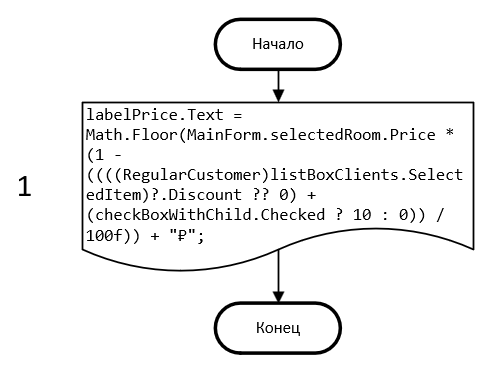


Рисунок 14 – Обновление суммы к оплате

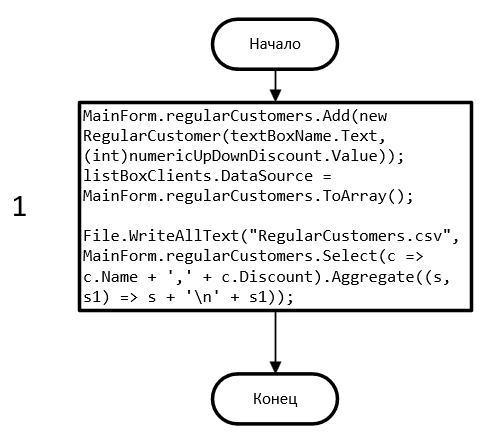


Рисунок 15 – Добавление клиента и сохранение файла

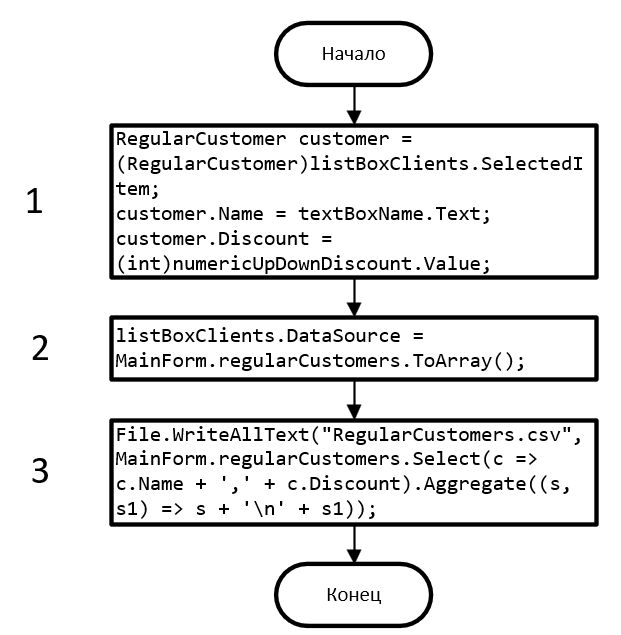


Рисунок 16 – Изменение клиента и сохранение файла

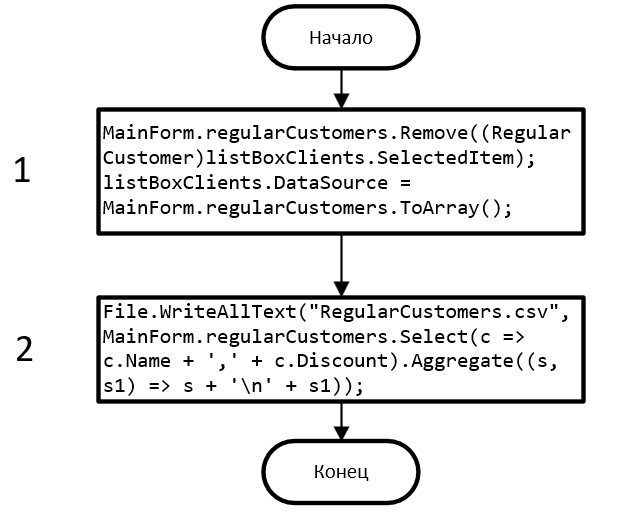


Рисунок 17 – Удаление клиента и сохранение файла

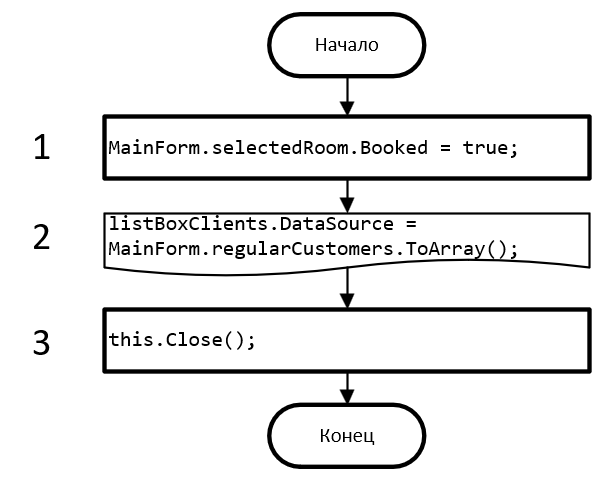


Рисунок 18 – Бронирование комнаты и закрытие формы

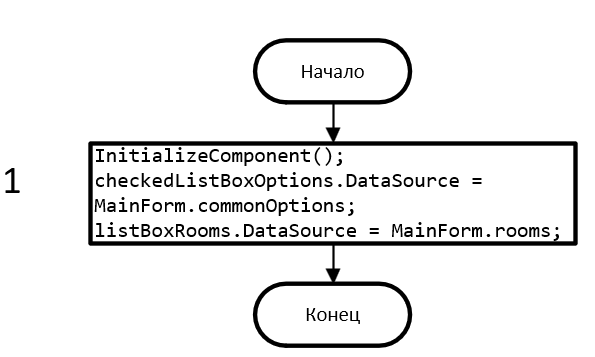


Рисунок 19 – Обновление списков номеров и опций

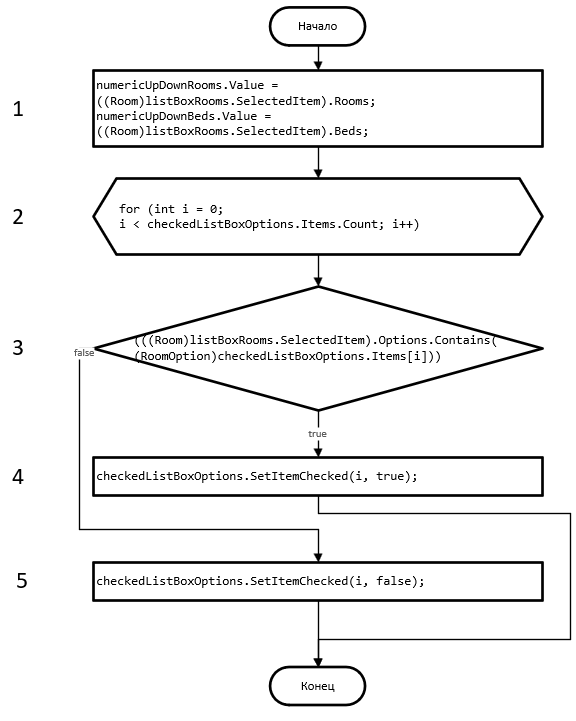


Рисунок 20 – Заполнение элементов формы данными о выбранном номере

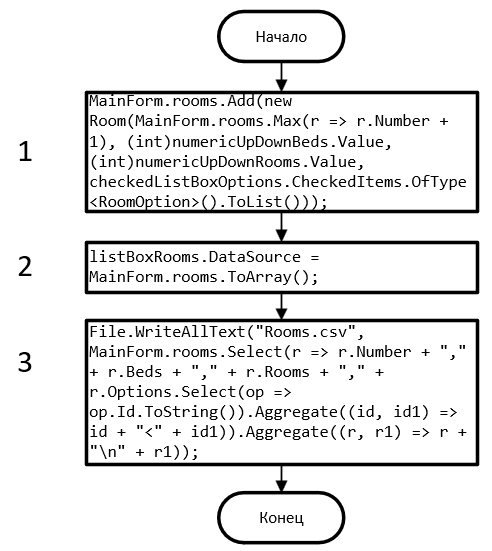


Рисунок 21 – Добавление номера и сохранение файла

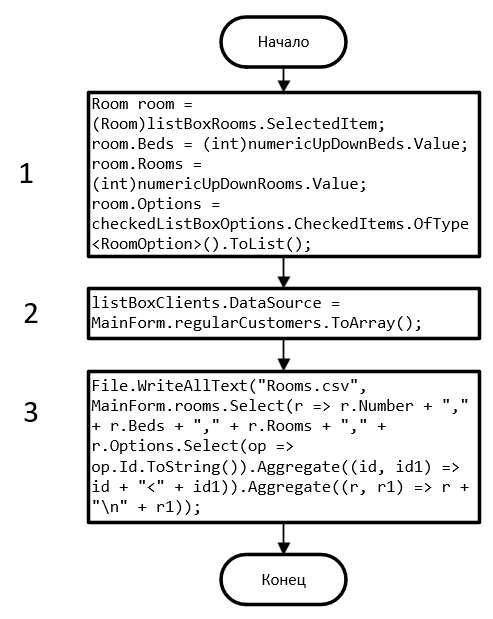


Рисунок 22 – Изменение номера и сохранение файла

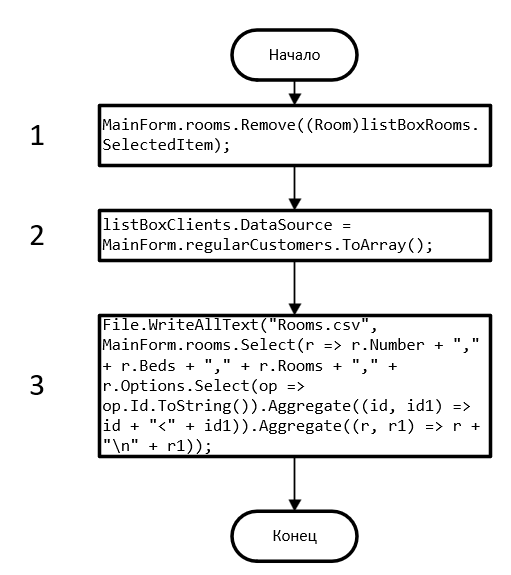


Рисунок 23 – Удаление номера и сохранение файла

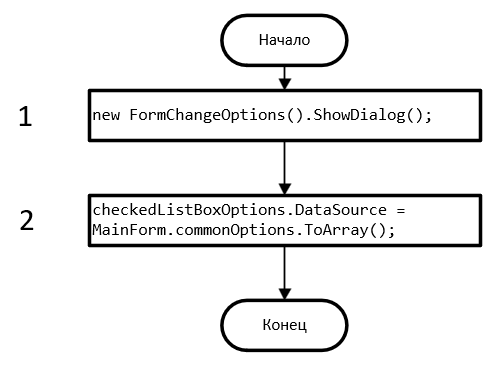


Рисунок 24 – Открытие формы изменения опций

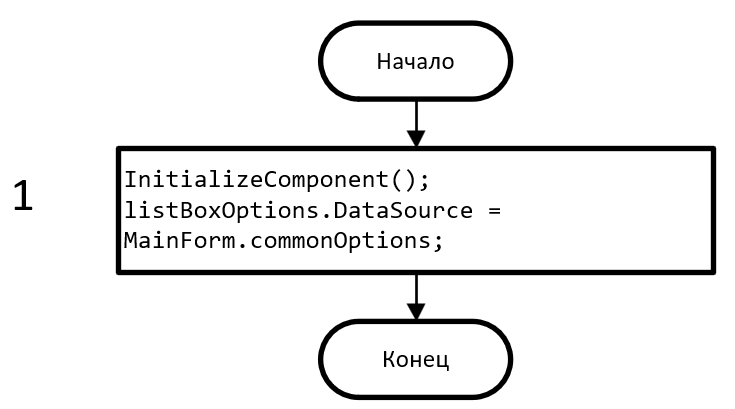


Рисунок 25 – Обновление списка опций

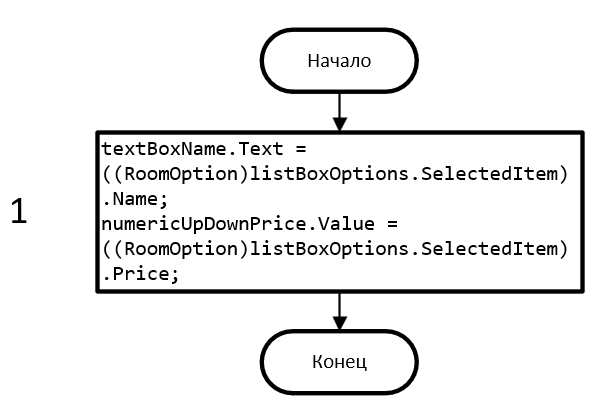


Рисунок 26 – Заполнение элементов формы данными о выбранной опции

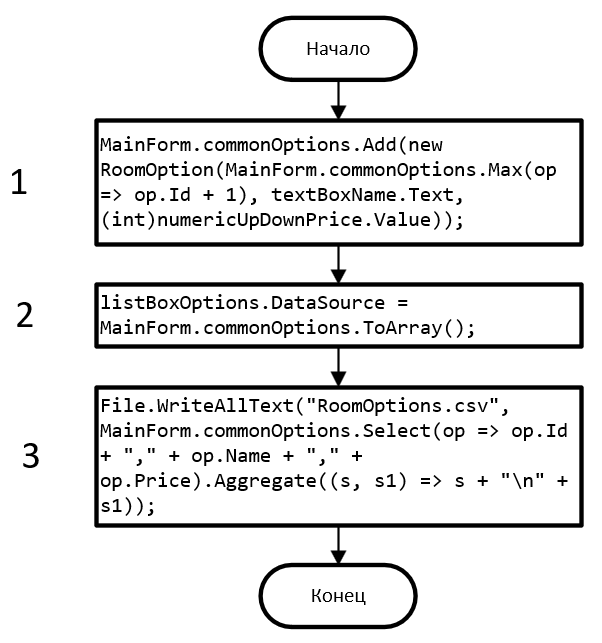


Рисунок 27 – Добавление опции и сохранение файла

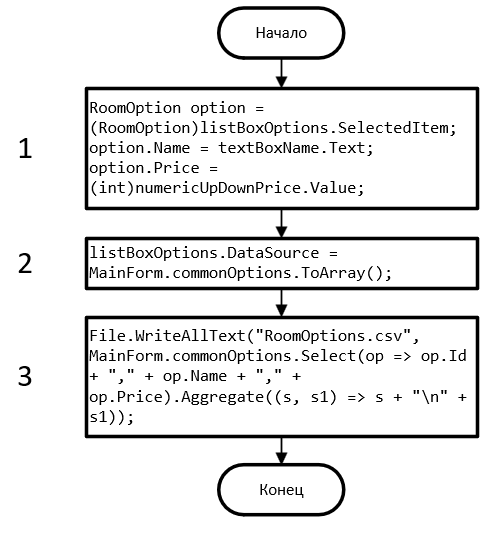


Рисунок 28 – Изменение опции и сохранение файла

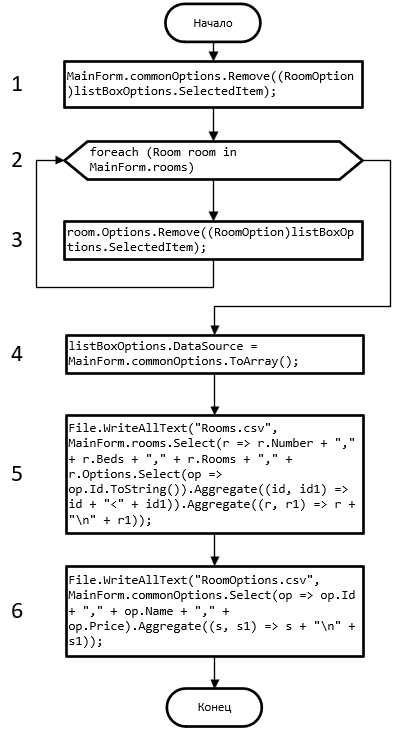


Рисунок 29 – Удаление опции и сохранение файла

1. **Подбор тестовых примеров**

На основании задания были определены две основные сущности программы: постоянный клиент и номер отеля. В целях упрощения расчетов и работы с сущностью «номер отеля» была введена дополнительная сущность – «удобство номера». В программе были созданы соответствующие классы: RegularCustomer, Room и RoomOption. Также для хранения данных этих классов были созданы и заполнены тестовыми данными csv-файлы:

Файл RegularCustomers.csv:

нет,0

Иванов И.И.,50

Андреев Ф.Н.,15

Анатольев Ф.А.,12

Музыченко Н.Б.,15

Петренко М.О.,7

Файл Rooms.csv:

1,2,1

2,2,1

3,2,1

4,2,1,Холодильник,Балкон

5,3,2,Минибар,Балкон

6,2,1,Холодильник,Балкон,Вид на море

7,4,3,Минибар,Балкон,Джакузи

Файл RoomOptions.csv:

Холодильник,500

Минибар,1500

Балкон,2000

Джакузи,1000

Телевизор,200

Вид на море,2000

1. **Листинг (код) составленного программного обеспечения**

**Файл MainForm.cs**

using System;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace LINQ\_querries

{

public partial class MainForm : Form

{

public static List<RoomOption> commonOptions;

public static List<Room> rooms;

public static List<RegularCustomer> regularCustomers;

public static Room selectedRoom;

public MainForm()

{

LoadCSV();

InitializeComponent();

checkedListBoxOptions.DataSource = commonOptions;

UpdateRooms();

}

public static RoomOption GetCommonOption(int id)

{

return commonOptions.FirstOrDefault(op => op.Id == id);

}

private void LoadCSV()

{

commonOptions = File.ReadAllText("RoomOptions.csv", System.Text.Encoding.UTF8).Split('\n').Select(s => { string[] s1 = s.Split(','); return new RoomOption(int.Parse(s1[0]), s1[1], int.Parse(s1[2])); }).ToList();

rooms = File.ReadAllText("Rooms.csv", System.Text.Encoding.UTF8).Split('\n').Select(s => { string[] s1 = s.Split(','); return new Room(int.Parse(s1[0]), int.Parse(s1[1]), int.Parse(s1[2]), (s1.Count() > 3) ? s1[3].Split('<').Select(opId => GetCommonOption(int.Parse(opId))).ToList() : null); }).ToList();

regularCustomers = File.ReadAllText("RegularCustomers.csv", System.Text.Encoding.UTF8).Split('\n').Select(s => { string[] s1 = s.Split(','); return new RegularCustomer(s1[0], int.Parse(s1[1])); }).ToList();

}

private void UpdateRooms()

{

List<Room> filtered = rooms.Where(r => r.Beds >= numericUpDownPlaces.Value && checkedListBoxOptions.CheckedItems.Cast<RoomOption>().All(op => r.Options.Contains(op)) && (!checkBoxFreeOnly.Checked || !r.Booked)).ToList();

listBoxRooms.DataSource = filtered;

label4.Text = $"Найдено: {filtered.Count()}, Свободно: {filtered.Count(r => !r.Booked)}";

}

private void checkBoxFreeOnly\_CheckedChanged(object sender, EventArgs e)

{

UpdateRooms();

}

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

{

UpdateRooms();

}

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

{

Room room = (Room)listBoxRooms.SelectedItem;

if (room == null)

{

MessageBox.Show("Комната не выбрана!");

return;

}

if (room.Booked)

{

room.Booked = false;

MessageBox.Show("Комната освобождена!");

}

else

MessageBox.Show("Комната уже свободна!");

UpdateRooms();

}

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

{

selectedRoom = (Room)listBoxRooms.SelectedItem;

if (selectedRoom == null)

{

MessageBox.Show("Комната не выбрана!");

return;

}

if (selectedRoom.Booked)

MessageBox.Show("Комната уже занята!");

else

{

new FormBook().ShowDialog();

}

UpdateRooms();

}

private void numericUpDownPlaces\_ValueChanged(object sender, EventArgs e)

{

UpdateRooms();

}

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

{

new ChangeRooms().ShowDialog();

UpdateRooms();

checkedListBoxOptions.DataSource = commonOptions.ToArray();

}

}

}

**Файл FormBook.cs**

using System;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace LINQ\_querries

{

public partial class FormBook : Form

{

public FormBook()

{

InitializeComponent();

listBoxClients.DataSource = MainForm.regularCustomers;

}

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

{

labelPrice.Text = Math.Floor(MainForm.selectedRoom.Price \* (1 - ((((RegularCustomer)listBoxClients.SelectedItem)?.Discount ?? 0) + (checkBoxWithChild.Checked ? 10 : 0)) / 100f)) + "₽";

textBoxName.Text = ((RegularCustomer)listBoxClients.SelectedItem).Name;

numericUpDownDiscount.Value = ((RegularCustomer)listBoxClients.SelectedItem).Discount;

}

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

{

MainForm.selectedRoom.Booked = true;

MessageBox.Show($"Комната забронирована!");

this.Close();

}

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

{

MainForm.regularCustomers.Add(new RegularCustomer(textBoxName.Text, (int)numericUpDownDiscount.Value));

listBoxClients.DataSource = MainForm.regularCustomers.ToArray();

File.WriteAllText("RegularCustomers.csv", MainForm.regularCustomers.Select(c => c.Name + ',' + c.Discount).Aggregate((s, s1) => s + '\n' + s1));

}

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

{

RegularCustomer customer = (RegularCustomer)listBoxClients.SelectedItem;

customer.Name = textBoxName.Text;

customer.Discount = (int)numericUpDownDiscount.Value;

listBoxClients.DataSource = MainForm.regularCustomers.ToArray();

File.WriteAllText("RegularCustomers.csv", MainForm.regularCustomers.Select(c => c.Name + ',' + c.Discount).Aggregate((s, s1) => s + '\n' + s1));

}

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

{

MainForm.regularCustomers.Remove((RegularCustomer)listBoxClients.SelectedItem);

listBoxClients.DataSource = MainForm.regularCustomers.ToArray();

File.WriteAllText("RegularCustomers.csv", MainForm.regularCustomers.Select(c => c.Name + ',' + c.Discount).Aggregate((s, s1) => s + '\n' + s1));

}

private void checkBoxWithChild\_CheckedChanged(object sender, EventArgs e)

{

labelPrice.Text = Math.Floor(MainForm.selectedRoom.Price \* (1 - ((((RegularCustomer)listBoxClients.SelectedItem)?.Discount ?? 0) + (checkBoxWithChild.Checked ? 10 : 0)) / 100f)) + "₽";

}

}

}

**Файл ChangeRooms.cs**

using System;

using System.Data;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace LINQ\_querries

{

public partial class ChangeRooms : Form

{

public ChangeRooms()

{

InitializeComponent();

checkedListBoxOptions.DataSource = MainForm.commonOptions;

listBoxRooms.DataSource = MainForm.rooms;

}

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

{

numericUpDownRooms.Value = ((Room)listBoxRooms.SelectedItem).Rooms;

numericUpDownBeds.Value = ((Room)listBoxRooms.SelectedItem).Beds;

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

if (((Room)listBoxRooms.SelectedItem).Options.Contains((RoomOption)checkedListBoxOptions.Items[i]))

checkedListBoxOptions.SetItemChecked(i, true);

else

checkedListBoxOptions.SetItemChecked(i, false);

}

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

{

MainForm.rooms.Add(new Room(MainForm.rooms.Max(r => r.Number + 1), (int)numericUpDownBeds.Value, (int)numericUpDownRooms.Value, checkedListBoxOptions.CheckedItems.OfType<RoomOption>().ToList()));

listBoxRooms.DataSource = MainForm.rooms.ToArray();

File.WriteAllText("Rooms.csv", MainForm.rooms.Select(r => r.Number + "," + r.Beds + "," + r.Rooms + "," + r.Options.Select(op => op.Id.ToString()).Aggregate((id, id1) => id + "<" + id1)).Aggregate((r, r1) => r + "\n" + r1));

}

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

{

Room room = (Room)listBoxRooms.SelectedItem;

room.Beds = (int)numericUpDownBeds.Value;

room.Rooms = (int)numericUpDownRooms.Value;

room.Options = checkedListBoxOptions.CheckedItems.OfType<RoomOption>().ToList();

listBoxRooms.DataSource = MainForm.rooms.ToArray();

File.WriteAllText("Rooms.csv", MainForm.rooms.Select(r => r.Number + "," + r.Beds + "," + r.Rooms + "," + r.Options.Select(op => op.Id.ToString()).Aggregate((id, id1) => id + "<" + id1)).Aggregate((r, r1) => r + "\n" + r1));

}

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

{

MainForm.rooms.Remove((Room)listBoxRooms.SelectedItem);

listBoxRooms.DataSource = MainForm.rooms.ToArray();

File.WriteAllText("Rooms.csv", MainForm.rooms.Select(r => r.Number + "," + r.Beds + "," + r.Rooms + "," + r.Options.Select(op => op.Id.ToString()).Aggregate((id, id1) => id + "<" + id1)).Aggregate((r, r1) => r + "\n" + r1));

}

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

{

new FormChangeOptions().ShowDialog();

checkedListBoxOptions.DataSource = MainForm.commonOptions.ToArray();

}

}

}

**Файл FormChangeOptions.cs**

using System;

using System.IO;

using System.Linq;

using System.Windows.Forms;

namespace LINQ\_querries

{

public partial class FormChangeOptions : Form

{

public FormChangeOptions()

{

InitializeComponent();

listBoxOptions.DataSource = MainForm.commonOptions;

}

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

{

textBoxName.Text = ((RoomOption)listBoxOptions.SelectedItem).Name;

numericUpDownPrice.Value = ((RoomOption)listBoxOptions.SelectedItem).Price;

}

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

{

MainForm.commonOptions.Add(new RoomOption(MainForm.commonOptions.Max(op => op.Id + 1), textBoxName.Text, (int)numericUpDownPrice.Value));

listBoxOptions.DataSource = MainForm.commonOptions.ToArray();

File.WriteAllText("RoomOptions.csv", MainForm.commonOptions.Select(op => op.Id + "," + op.Name + "," + op.Price).Aggregate((s, s1) => s + "\n" + s1));

}

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

{

RoomOption option = (RoomOption)listBoxOptions.SelectedItem;

option.Name = textBoxName.Text;

option.Price = (int)numericUpDownPrice.Value;

listBoxOptions.DataSource = MainForm.commonOptions.ToArray();

File.WriteAllText("RoomOptions.csv", MainForm.commonOptions.Select(op => op.Id + "," + op.Name + "," + op.Price).Aggregate((s, s1) => s + "\n" + s1));

}

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

{

MainForm.commonOptions.Remove((RoomOption)listBoxOptions.SelectedItem);

foreach (Room room in MainForm.rooms)

room.Options.Remove((RoomOption)listBoxOptions.SelectedItem);

listBoxOptions.DataSource = MainForm.commonOptions.ToArray();

File.WriteAllText("Rooms.csv", MainForm.rooms.Select(r => r.Number + "," + r.Beds + "," + r.Rooms + "," + r.Options.Select(op => op.Id.ToString()).Aggregate((id, id1) => id + "<" + id1)).Aggregate((r, r1) => r + "\n" + r1));

File.WriteAllText("RoomOptions.csv", MainForm.commonOptions.Select(op => op.Id + "," + op.Name + "," + op.Price).Aggregate((s, s1) => s + "\n" + s1));

}

}

}

**Файл RegularCustomer.cs**

namespace LINQ\_querries

{

public class RegularCustomer

{

public string Name { get; set; }

public int Discount { get; set; }

public RegularCustomer(string name, int discount)

{

Name = name;

Discount = discount;

}

public override string ToString()

{

return $"{Name} со скидкой {Discount}%" ;

}

}

}

**Файл Room.cs**

using System.Collections.Generic;

using System.Linq;

namespace LINQ\_querries

{

public class Room

{

private const int basePrice = 500;

public List<RoomOption> Options = new List<RoomOption>() { MainForm.GetCommonOption(5) };

public readonly int Number;

public int Beds { get; set; }

public int Rooms { get; set; }

public int Price => ((basePrice + Options.Sum(op => op.Price)) \* Rooms) + basePrice \* Beds;

public bool Booked;

public Room(int number, int beds = 2, int rooms = 1, List<RoomOption> options = null)

{

Number = number;

Beds = beds;

Rooms = rooms;

if (options != null)

Options = options;

}

public override string ToString()

{

return $"№{Number}. {Beds}-местный, {Rooms}-комнатный. Цена: {Price}₽. Удобства: " + Options.Select(op => op.Name).Aggregate((n1, n2) => n1 + ", " + n2) + ". " +(Booked ? "Занят" : "Свободен");

}

}

}

**Файл RoomOption.cs**

namespace LINQ\_querries

{

public class RoomOption

{

public int Id;

public string Name { get; set; }

public int Price { get; set; }

public RoomOption(int id, string name, int price)

{

Id = id;

Name = name;

Price = price;

}

public override string ToString()

{

return $"{Name} за {Price}₽";

}

}

}

1. **Графический пользовательский интерфейс программного обеспечения и его описание**

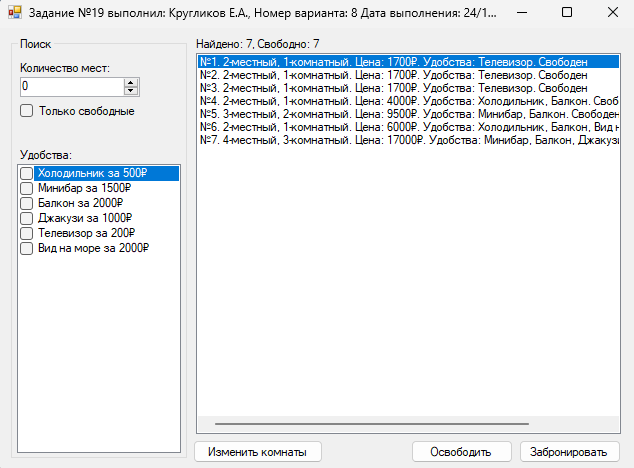


Рисунок 30 – Интерфейс главной формы

В левой части формы находятся фильтры для поиска нужного номера: минимальное количество мест в номере, чекбокс для показа только свободных номеров, а также список необходимых удобств.

В правой части формы находится список всех номеров, соответствующих критериям с полным описанием номера. Также внизу справа находятся кнопки для брони/освобождения выбранного номера.

Внизу слева под списком комнат находится кнопка «Изменить комнаты» для открытия формы с редактированием данных о номерах.

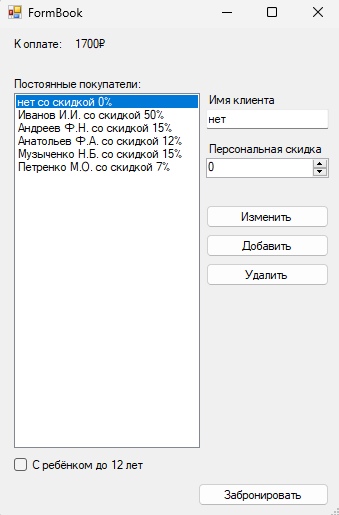


Рисунок 31 – Интерфейс формы бронирования

В левой части находится список постоянных покупателей и чекбокс «С ребенком до 12 лет». При выборе постоянного покупателя из списка при оплате рассчитывается соответствующая скидка. Также при отметке чекбокса есть дополнительная скидка 10%. Сумма после применения скидок выводится в верхней части формы.

В правой части формы находятся элементы управления для изменения информации о постоянных клиентах.

При нажатии кнопки «Изменить комнаты» в главной форме открывается формы редактирования комнат:

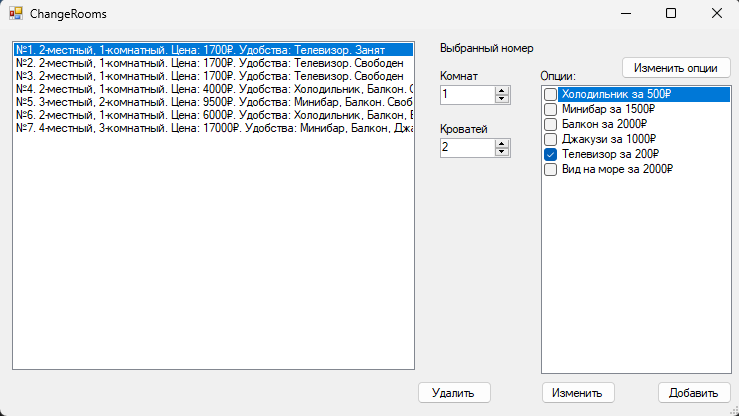


Рисунок 32 – Интерфейс формы изменения комнат

В левой части формы находится список существующих комнат.

В правой части формы находятся элементы управления, позволяющие редактировать информацию о номерах отеля.

Также справа над списком опций находится кнопка «Изменить опции», при нажатии на которую открывается форма изменения опций:

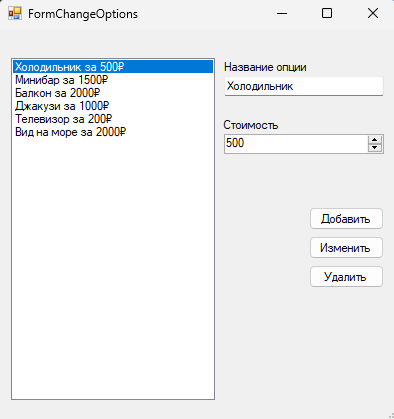


Рисунок 33 – Интерфейс формы изменения опций

В левой части находится список опций, доступных в программе.

В правой части формы находятся элементы управления, позволяющие редактировать информацию об опциях номеров.

1. **Подтверждение соответствия графического пользовательского интерфейса требованиям к оформлению**

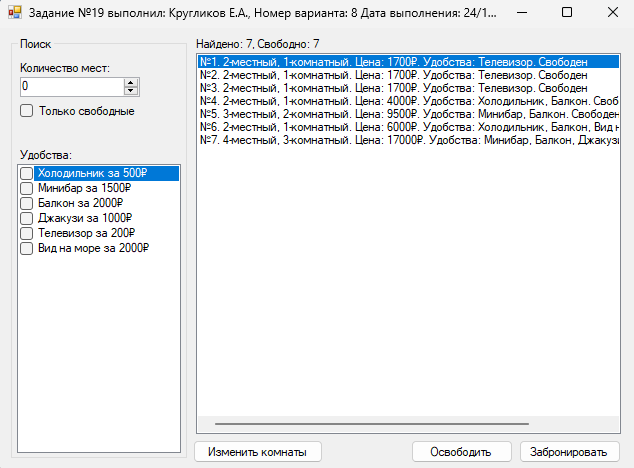


Рисунок 34 – Подтверждение соответствия графического пользовательского интерфейса требованиям к оформлению.

1. Экранный список в правой части формы и настройки фильтров слева позволяют предоставлять (и бронировать) посетителям номер в гостинице в зависимости от их потребностей (количества мест для размещения группы гостей и т.д.);
2. Они же позволяют показывать количество свободных (занятых) мест по категории номеров указанного типа;
3. Экранный список в левой части формы бронирования позволяет показывать список постоянных посетителей и предоставляемую им скидку;
4. В правом экранном списке выведены все номера, что позволяет показывать информацию по указанному номеру (количество мест, комнат, наличие холодильника, телевизора, бара и т.д.).

Таким образом, разработанный графический интерфейс реализует все требования, изложенные в задании к разработке.

1. **Расчёт тестовых примеров с использованием составленного программного обеспечения**

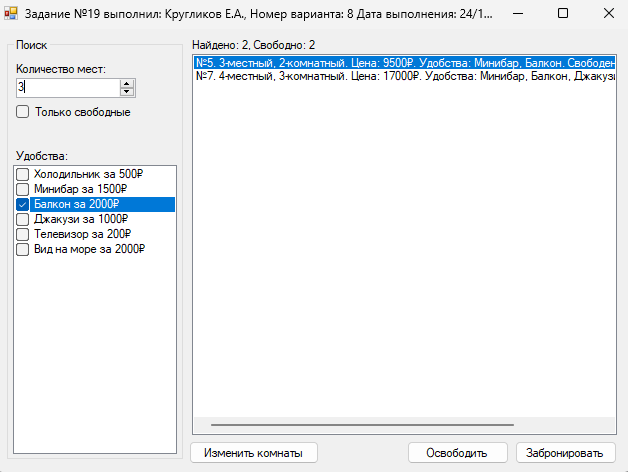


Рисунок 5 – Демонстрация работы фильтра по количеству мест и необходимым удобствам

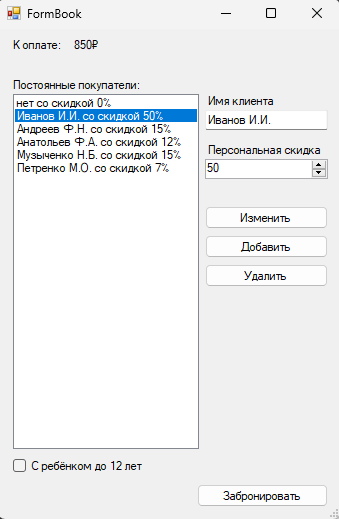
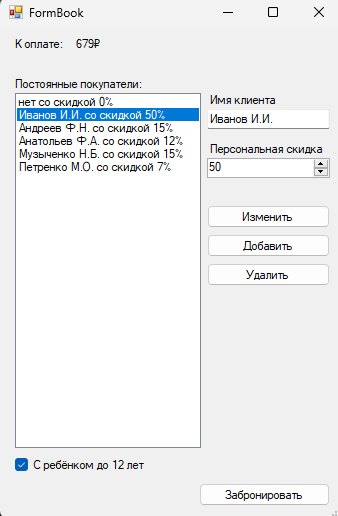
 

Рисунок 4 – Форма бронирования с суммой оплаты при выборе номера №1 и выборе покупателя «Иванов И.И.» (скидка 50%) с отметкой «С ребенком до 12 лет» (справа) и без (слева).

1. **Формулировка вывода о проделанной работе**

Освоил навыки подготовки наборов данных по нечётко сформулированному описанию предметной области при учёте особенностей запросов, отклики на которые необходимо предоставлять заявленным (потенциальным) пользователям программного обеспечения.

Научился выделять основные и второстепенные сущности и логику их взаимодействия в предметной области. Освоил навыки переноса этих сущностей и взаимодействий в логику программы.