Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет «Московский институт электронной техники»

Институт системной и программной инженерии и информационных технологий

**Курсовая работа**

по дисциплине«**Компьютерная практика**»

на тему:

**«Информационно-справочная система**

**учета жителей гостиницы»**

**Выполнил**: студент группы П-22

Д.В. Поздняков

**Проверил**: А.И. Квач

Оглавление

[Введение 3](#_Toc59895080)

[1 Функциональные требования, предъявляемые к программной системе 4](#_Toc59895081)

[1.1 Требование к функциям (задачам) 4](#_Toc59895082)

[1.2 Требования к видам обеспечения 4](#_Toc59895083)

[2 Краткое описание метода решения поставленной задачи 5](#_Toc59895084)

[3 Описание структуры данных 6](#_Toc59895085)

[4 Алгоритмы работы программной системы 7](#_Toc59895086)

[5 Текст программы 10](#_Toc59895087)

[6 Примеры работы системы 19](#_Toc59895088)

[7 Руководство пользователя по работе с программной системой 23](#_Toc59895089)

[7.1 Установка системы 23](#_Toc59895090)

[7.2 Работа в системе 23](#_Toc59895091)

[Заключение 24](#_Toc59895092)

[Список иллюстраций 25](#_Toc59895093)

[Список таблиц 25](#_Toc59895094)

**Бланк задания на курсовую работу**

УТВЕРЖДАЮ:

Зам. зав. кафедрой\_\_\_\_\_\_\_\_\_\_ (Н.Ю. Соколова)

«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2020 г.

**Задание № C-2**

**на выполнение курсовой работы**

**по дисциплине «Компьютерная практика»**

Студент группы **П-22** Направление: «**Прикладная информатика**»

Фамилия: **Поздняков** Имя: **Даниил** Отчество: **Васильевич**

Руководитель курсовой работы: Квач Артем Игоревич ([creeone@mail.ru](mailto:creeone@mail.ru))

Срок выполнения работы: с «01» ноября 2020 г. по «29» декабря 2020 г.

1. **Тема курсовой работы:**

**«Разработка информационно- справочной системы учета жителей гостиницы».**

Система должна предоставлять возможность:

- формировать список жителей гостиницы, добавляя в имеющийся или создавая новый список,

- просматривать список всех проживающих,

- осуществлять поиск свободных номеров,

- получать информацию о проживающих, которые должны выехать на определенную дату,

1. **Требования, предъявляемые к разработке системы:**

* результаты выполнения программы должны отображаться в табличном виде;
* среда программирования – Visual Studio 2005, 2008.

1. **Пояснительная записка (ПЗ) должна содержать следующие разделы основной части:**

* описание функциональных требований, предъявляемых к программной системе (техническое задание –ТЗ-на разработку),
* краткое описание метода решения поставленной задачи,
* описание структуры данных,
* алгоритм работы программной системы,
* текст программы,
* скриншоты работы системы при различных данных,
* руководство пользователя по работе с программной системой (установка системы, работа в системе).

К защите предоставляются следующие материалы

* 1. Пояснительная записка (см. Метод. указания).
  2. Компьютерные версии: ПЗ (см. Метод. указания), доклад, презентация доклада, работающая программа.

**Допуск к защите** курсовой работы осуществляется только после предварительной сдачи преподавателю всего набора файлов и утвержденной пояснительной записки.

# Введение

Целью данной работы является закрепление знаний, полученных при изучении дисциплин «Программирование на языке высокого уровня С++» и «Компьютерная практика», а также обучение их применения на практике путём разработки информационно-справочной системы учета клиентов рекламной компании.

Задачами работы являются:

* анализ предметной области, написание технического задания;
* разработка схем алгоритмов;
* написание и отладка программы.

# Назначение и цель создания (развития) системы

Система предназначена для ведения информационно-справочной системы учета жителей в гостинице

Цель создания системы является:

-упрощение учета свободных помещений и жителей во время пребывания.

# **Характеристики объекта автоматизации**

Ежедневно через гостиницу проходит большое количество жителей, которых очень важно учитывать, чтобы не потеряться в финансах и оперативно анализировать свободные и занятые номера.

# Требования к системе

Пользователями системы являются: руководители предприятия, администраторы.

## Требование к функциям (задачам)

Система должна предоставлять возможность:

- формировать список жителей гостиницы, добавляя в имеющийся или создавая новый список,

- просматривать список всех проживающих,

- осуществлять поиск свободных номеров,

- получать информацию о проживающих, которые должны выехать на определенную дату,

- получать информацию о долгожителях гостиницы

## Требование к видам обеспечения

Для реализации системы требуется ОС Windows 7

Система должна быть реализована в среде Visual Studio 2005 и выше, на языке программирования С++.

Результаты выполнения программы должны отображаться в табличном виде.

Приложение должно максимально соответствовать интерфейсу Windows.

Рабочее место сотрудников рекламной компании должно быть оборудовано компьютером с характеристиками не ниже: процессор с тактовой частотой 1,8 ГГц, 2 ГБ ОЗУ, 5 ГБ свободного дискового пространства, дисплей с разрешением 1280х720.Результаты выполнения программы должны отображаться в табличном виде.

# Краткое описание метода решения поставленной задачи

Реализация системы осуществлялась с использованием СУБД Microsoft Access и Windows Forms.

Причины выбора вышеуказанного метода реализации:

* Удобство создания табличной структуры данных в СУБД Microsoft Access;
* Возможность создания приложения, максимально соответствующего интерфейсу ОС Windows и интуитивно понятного пользователю, при помощи Windows Forms;
* Возможность соединения СУБД и Windows Forms.

# Описание структуры данных

Структура данных, реализованная в СУБД Microsoft Access в сущности «clients», представлена в Таблица 1.

Таблица 1 – Структура данных

|  |  |  |
| --- | --- | --- |
| **Атрибут сущности** | **Тип данных** | **Описание** |
| ID | Числовой | Уникальный номер постояльца, присваиваемый гостиницей |
| Name | Текстовый | ФИО постояльца |
| Pass | Числовой | Паспорт постояльца |
| City | Текстовый | Город проживания |
| DataConc | Текстовый | Дата заселения |
| DataExp | Текстовый | Дата выселения |
| Breakfast | Логический | Заказал ли постоялец завтрак |
| Num\_room | Числовой | Номер комнаты |
| Phone\_number | Текстовый | Номер телефона постояльца |

Реализация структуры данных в СУДБ Microsoft Access представлена на рисунке 1.

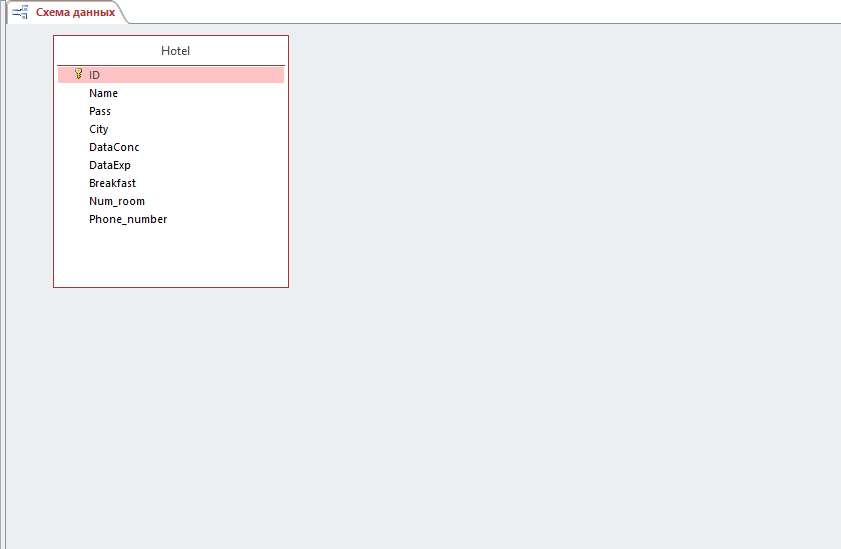


Рисунок 1 – Реализация структуры данных в СУДБ Microsoft Access

# Алгоритмы работы программной системы

Общий алгоритм работы программы представлен на Рисунок 2.



Рисунок 2 – Общий алгоритм работы программы

Общий вид алгоритма работы функций просмотра всех клиентов и функции просмотра постоянных клиентов компании представлена на Рисунок 3. При реализации функций отличия в алгоритме появляются на этапе запроса к БД.



Рисунок 3 – Общий вид функции просмотра всех клиентов и долгожителей

Алгоритм добавления данных о клиенте представлен на Рисунок 4.



Рисунок 4 – Алгоритм добавления данных

Алгоритм удаления данных о клиенте представлен на Рисунок 5.



Рисунок 5 – Алгоритм удаления данных

Общий вид алгоритма работы функций поиска клиентов по округу и сроку истечения договора представлен на Рисунок 6. При реализации функций отличия в алгоритме появляются на этапе запроса к БД.



Рисунок 6 – Общий вид функций поиска клиентов по округу и сроку

# Текст программы

#include "MyForm.h"

using namespace System;

using namespace System::Windows::Forms;

using namespace System::Data::OleDb;

[STAThread]

int main(array<String^>^ arg) //основа для формы

{

Application::EnableVisualStyles();

Application::SetCompatibleTextRenderingDefault(false);

Accuracy::MyForm form;

Application::Run(% form);

}

//Список всех клиентов

System::Void Accuracy::MyForm::button\_clients\_Click(System::Object^ sender, System::EventArgs^ e)

{

dataGridView1->Rows->Clear();

//Подключение к БД

String^ connectionString = "provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

OleDbConnection^ dbConnection = gcnew OleDbConnection(connectionString);

//Запрос к БД

dbConnection->Open(); //открытие соединения

String^ query = "SELECT \* FROM [Hotel]"; //запрос

OleDbCommand^ dbComand = gcnew OleDbCommand(query, dbConnection); //команда

OleDbDataReader^ dbReader = dbComand->ExecuteReader(); //считывание данных

//Проверка данных

if (dbReader->HasRows == false)

{

MessageBox::Show("Ошибка считывания данных", "Ошибка");

}

else

{

while (dbReader->Read())

{

dataGridView1->Rows->Add(dbReader["ID"], dbReader["Name"], dbReader["Pass"], dbReader["City"], dbReader["DataConc"], dbReader["DataExp"], dbReader["Breakfast"], dbReader["Num\_room"], dbReader["Phone\_number"]);

}

}

//Закрытие соединения

dbReader->Close();

dbConnection->Close();

return System::Void();

}

//Список долгожителей

System::Void Accuracy::MyForm::button\_regular\_Click(System::Object^ sender, System::EventArgs^ e)

{

dataGridView1->Rows->Clear();

//Подключение к БД

String^ connectionString = "provider=Microsoft.Jet.OLEDB.4.0;Data Source=hotel.mdb";

OleDbConnection^ dbConnection = gcnew OleDbConnection(connectionString);

//Запрос к БД

dbConnection->Open(); //открытие соединения

String^ query = "SELECT \* FROM Hotel WHERE DATEDIFF('d',DataConc,DataExp) >= 5"; //запрос

OleDbCommand^ dbComand = gcnew OleDbCommand(query, dbConnection); //команда

OleDbDataReader^ dbReader = dbComand->ExecuteReader(); //считывание данных

//Проверка данных

if (dbReader->HasRows == false)

{

MessageBox::Show("Ошибка считывания данных", "Ошибка");

}

else

{

while (dbReader->Read())

{

dataGridView1->Rows->Add(dbReader["ID"], dbReader["Name"], dbReader["Pass"], dbReader["City"], dbReader["DataConc"], dbReader["DataExp"], dbReader["Breakfast"], dbReader["Num\_room"], dbReader["Phone\_number"]);

}

}

//Закрытие соединения

dbReader->Close();

dbConnection->Close();

return System::Void();

}

//Поиск по дате свободного номера

System::Void Accuracy::MyForm::button\_empty\_Click(System::Object^ sender, System::EventArgs^ e)

{

//Подключение к БД

String^ connectionString = "provider=Microsoft.Jet.OLEDB.4.0;Data Source=Hotel.mdb";

OleDbConnection^ dbConnection = gcnew OleDbConnection(connectionString);

//Запрос к БД

dbConnection->Open(); //открытие соединения

String^ query = "SELECT Num\_room FROM Hotel WHERE NOT Num\_room = ANY(SELECT Num\_room FROM Hotel WHERE ID = ANY(SELECT ID FROM Hotel WHERE '"+textBox\_empty->Text+"' BETWEEN DataConc AND DataExp)) GROUP BY Num\_room"; //запрос

OleDbCommand^ dbComand = gcnew OleDbCommand(query, dbConnection); //команда

OleDbDataReader^ dbReader = dbComand->ExecuteReader(); //считывание данных

//Проверка данных

if (dbReader->HasRows == false)

{

MessageBox::Show("На данное число нет свободных номеров. \nПроверьте правильность ввода данных.", "Результаты поиска");

}

else

{

//dataGridView1->Rows->Clear();

String^ txt;

int kol\_elem=0;

while (dbReader->Read())

{

kol\_elem++;

txt =txt+"Номер "+dbReader["Num\_room"]->ToString() +"\n";

//dataGridView1->Rows->Add(dbReader["Num\_room"]);

}

MessageBox::Show("На данное число свободны "+kol\_elem+" номера: \n" + txt , "Результаты поиска");

}

//Закрытие соединения

dbReader->Close();

dbConnection->Close();

return System::Void();

}

//Поиск по дате выселения

System::Void Accuracy::MyForm::button\_date\_Click(System::Object^ sender, System::EventArgs^ e)

{

//Подключение к БД

String^ connectionString = "provider=Microsoft.Jet.OLEDB.4.0;Data Source=Hotel.mdb";

OleDbConnection^ dbConnection = gcnew OleDbConnection(connectionString);

//Запрос к БД

dbConnection->Open(); //открытие соединения

String^ query = "SELECT \* FROM Hotel WHERE DataExp='" + textBox\_date->Text + "'"; //запрос

OleDbCommand^ dbComand = gcnew OleDbCommand(query, dbConnection); //команда

OleDbDataReader^ dbReader = dbComand->ExecuteReader(); //считывание данных

//Проверка данных

if (dbReader->HasRows == false)

{

MessageBox::Show("На данную дату нет ни одного номера. \nПроверьте правильность ввода данных.", "Результаты поиска");

}

else

{

dataGridView1->Rows->Clear();

while (dbReader->Read())

{

dataGridView1->Rows->Add(dbReader["ID"], dbReader["Name"], dbReader["Pass"], dbReader["City"], dbReader["DataConc"], dbReader["DataExp"], dbReader["Breakfast"], dbReader["Num\_room"], dbReader["Phone\_number"]);

}

}

//Закрытие соединения

dbReader->Close();

dbConnection->Close();

return System::Void();

}

//Добавление

System::Void Accuracy::MyForm::button\_add\_Click(System::Object^ sender, System::EventArgs^ e)

{

//Выбор строки для добавления

if (dataGridView1->SelectedRows->Count != 1)

{

MessageBox::Show("Выберите одну строку для добавления", "Внимание");

return;

}

//Получение индекса выбранной строки

int index = dataGridView1->SelectedRows[0]->Index;

//Проверка данных

if (dataGridView1->Rows[index]->Cells[0]->Value == nullptr ||

dataGridView1->Rows[index]->Cells[1]->Value == nullptr ||

dataGridView1->Rows[index]->Cells[2]->Value == nullptr ||

dataGridView1->Rows[index]->Cells[3]->Value == nullptr ||

dataGridView1->Rows[index]->Cells[4]->Value == nullptr ||

dataGridView1->Rows[index]->Cells[5]->Value == nullptr ||

dataGridView1->Rows[index]->Cells[6]->Value == nullptr ||

dataGridView1->Rows[index]->Cells[7]->Value == nullptr ||

dataGridView1->Rows[index]->Cells[8]->Value == nullptr)

{

MessageBox::Show("Были введены не все данные", "Ошибка");

return;

}

//Считывание данных

String^ ID = dataGridView1->Rows[index]->Cells[0]->Value->ToString();

String^ Name = dataGridView1->Rows[index]->Cells[1]->Value->ToString();

String^ Pass = dataGridView1->Rows[index]->Cells[2]->Value->ToString();

String^ City = dataGridView1->Rows[index]->Cells[3]->Value->ToString();

String^ DataConc = dataGridView1->Rows[index]->Cells[4]->Value->ToString();

String^ DataExp = dataGridView1->Rows[index]->Cells[5]->Value->ToString();

String^ Breakfast = dataGridView1->Rows[index]->Cells[6]->Value->ToString();

String^ Num\_room = dataGridView1->Rows[index]->Cells[7]->Value->ToString();

String^ Phone\_number = dataGridView1->Rows[index]->Cells[8]->Value->ToString();

//Подключение к БД

String^ connectionString = "provider=Microsoft.Jet.OLEDB.4.0;Data Source=Hotel.mdb";

OleDbConnection^ dbConnection = gcnew OleDbConnection(connectionString);

//Запрос к БД

dbConnection->Open(); //открытие соединения

String^ query = "INSERT INTO [Hotel] VALUES ("+ ID +", '"+ Name +"'," + Pass + ", '" + City + "', '" + DataConc + "', '" + DataExp + "'," + Breakfast + "," + Num\_room + ", '" + Phone\_number + "')"; //запрос

OleDbCommand^ dbComand = gcnew OleDbCommand(query, dbConnection); //команда

//Выполнение запроса

if (dbComand->ExecuteNonQuery() !=1) MessageBox::Show("Ошибка выполнения запроса", "Ошибка");

else MessageBox::Show("Данные добавлены", "Готово");

//Закрытие соединения

dbConnection->Close();

return System::Void();

}

//Удаление клиента

System::Void Accuracy::MyForm::button\_delete\_Click(System::Object^ sender, System::EventArgs^ e)

{

//Выбор строки для удаления

if (dataGridView1->SelectedRows->Count != 1)

{

MessageBox::Show("Выберите одну строку для удаления", "Внимание");

return;

}

//Получение индекса выбранной строки

int index = dataGridView1->SelectedRows[0]->Index;

//Считывание данных

String^ ID = dataGridView1->Rows[index]->Cells[0]->Value->ToString();

//Подключение к БД

String^ connectionString = "provider=Microsoft.Jet.OLEDB.4.0;Data Source=Hotel.mdb";

OleDbConnection^ dbConnection = gcnew OleDbConnection(connectionString);

//Запрос к БД

dbConnection->Open(); //открытие соединения

String^ query = "DELETE FROM Hotel WHERE ID =" + ID; //запрос

OleDbCommand^ dbComand = gcnew OleDbCommand(query, dbConnection); //команда

//Выполнение запроса

if (dbComand->ExecuteNonQuery() != 1) MessageBox::Show("Ошибка выполнения запроса", "Ошибка");

else MessageBox::Show("Данные были удалены", "Готово");

//Закрытие соединения

dbConnection->Close();

return System::Void();

}

//Пункты меню

System::Void Accuracy::MyForm::оПрограммеToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("Система предназначена для ведения учета клиентов гостиницы и получения информации по запросам пользователей.\n \nПрограмма может: \n-формировать список жителей гостиницы, добавляя в имеющийся или создавая новый список,\n-просматривать список всех проживающих,\n-осуществлять поиск свободных номеров,\n-получать информацию о проживающих, которые должны выехать на определенную дату,\n-получать информацию о долгожителях гостиницы.","Возможности программы");

return System::Void();

}

System::Void Accuracy::MyForm::просмотрСпискаКлиентовToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("Для просмотра списка всех клиентов компании нажмите кнопку «Все клиенты»","Просмотр списка клиентов");

return System::Void();

}

System::Void Accuracy::MyForm::просмотрДолгожителейToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("Для просмотра списка постоянных клиентов компании нажмите кнопку «Долгожители»", "Просмотр списка Долгожителей");

return System::Void();

}

System::Void Accuracy::MyForm::добавлениеНовогоКлиентаToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("Для добавления нового клиента необходимо: \n 1) Ввести в пустую строку таблицы все данные о клиенте \n 2) Выделить всю строку введенных значений \n 3) Нажать кнопку «Добавить клиента»", "Добавление нового клиента");

return System::Void();

}

System::Void Accuracy::MyForm::удалениеКлиентаToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("Для удаления клиента необходимо: \n 1) Выделить всю строку, которую необходимо удалить \n 2) Нажать кнопку «Удалить клиента» \n 3) Снова нажать кнопку «Все клиенты» для обновления списка", "Удаление клиента");

return System::Void();

}

System::Void Accuracy::MyForm::поискСвободногоНомераToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("Для поиска свободного номера необходимо: \n 1) Ввести дату в формате «ХХ.ХХ.ХХХХ» в строку под надписью «Введите дату заселения» \n 2) Нажать кнопку «Поиск номера»", "Поиск номера");

return System::Void();

}

System::Void Accuracy::MyForm::ПоискДатыВыселенияToolStripMenuItem\_Click(System::Object^ sender, System::EventArgs^ e)

{

MessageBox::Show("Для поиска клиентов с выселением на определенную дату, необходимо: \n 1) Ввести дату в формате «ХХ.ХХ.ХХХХ» в строку под надписью «Введите дату выселения» \n 2) Нажать кнопку «Поиск по дате»", "Поиск по дате выселения");

return System::Void();

}

# Примеры работы системы

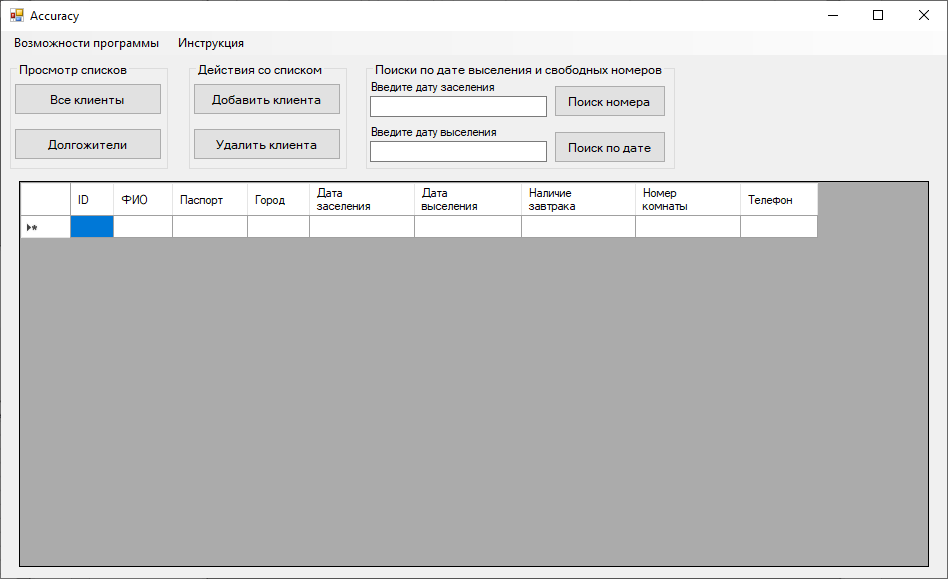


Рисунок 7 – Начальный экран программы

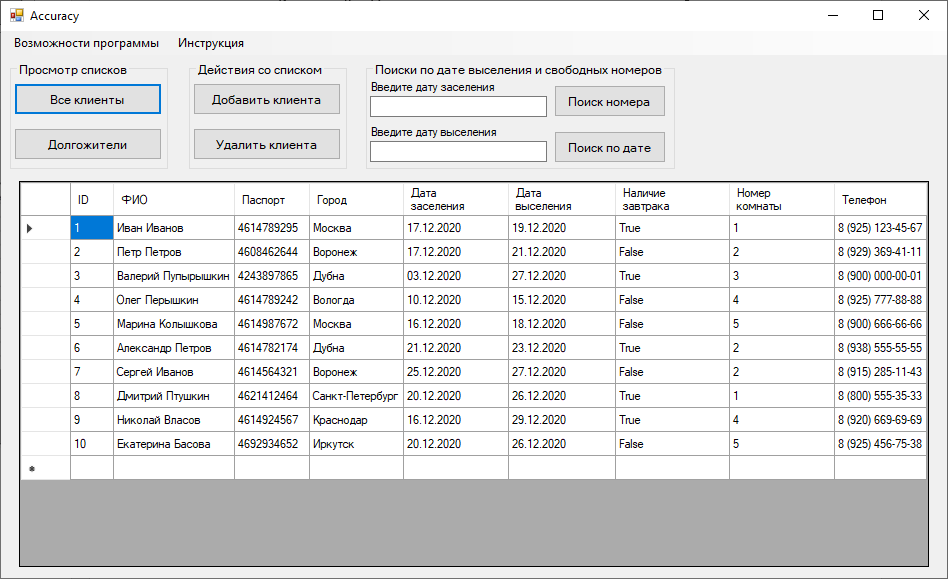


Рисунок 8 – Просмотр всех клиентов гостиницы

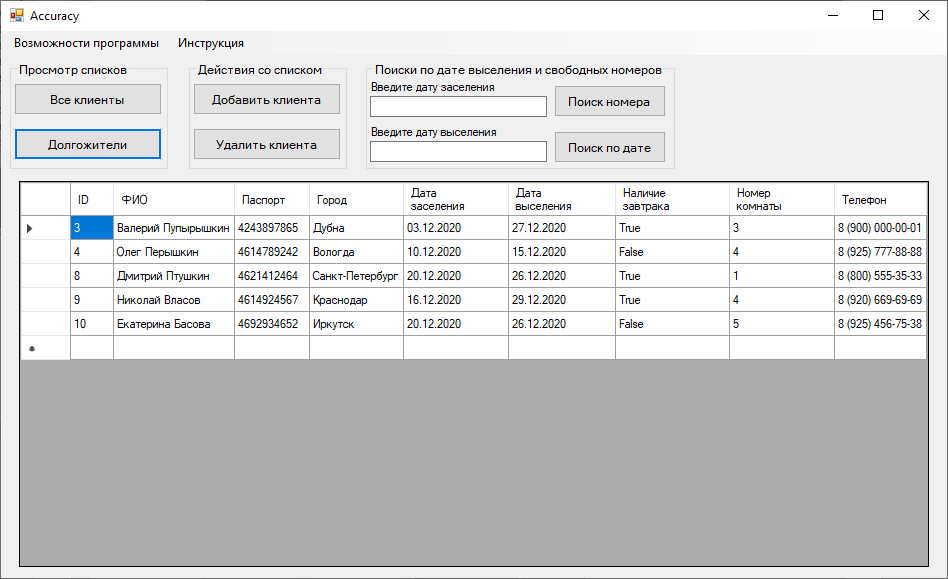


Рисунок 9 – Просмотр долгожителей гостиницы

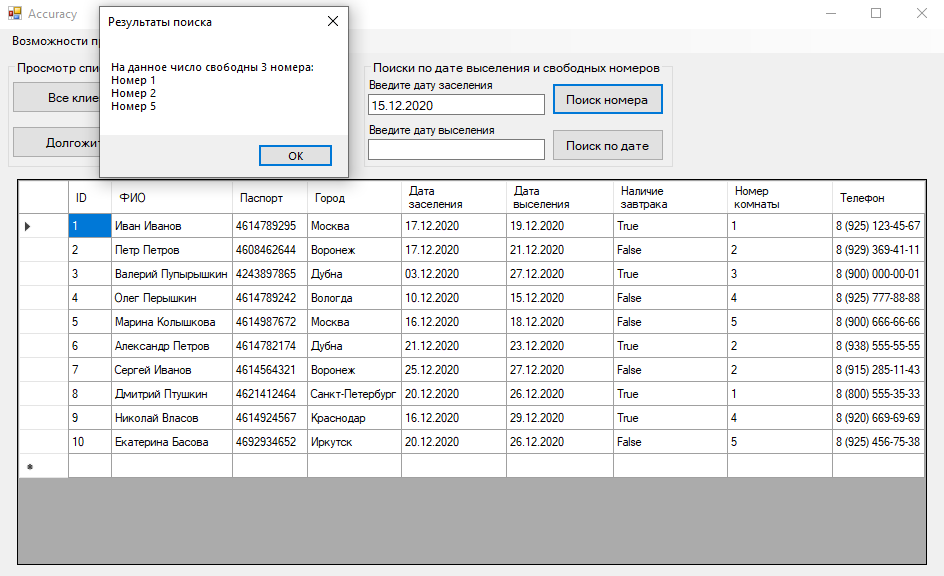


Рисунок 10 – Поиск свободных номеров

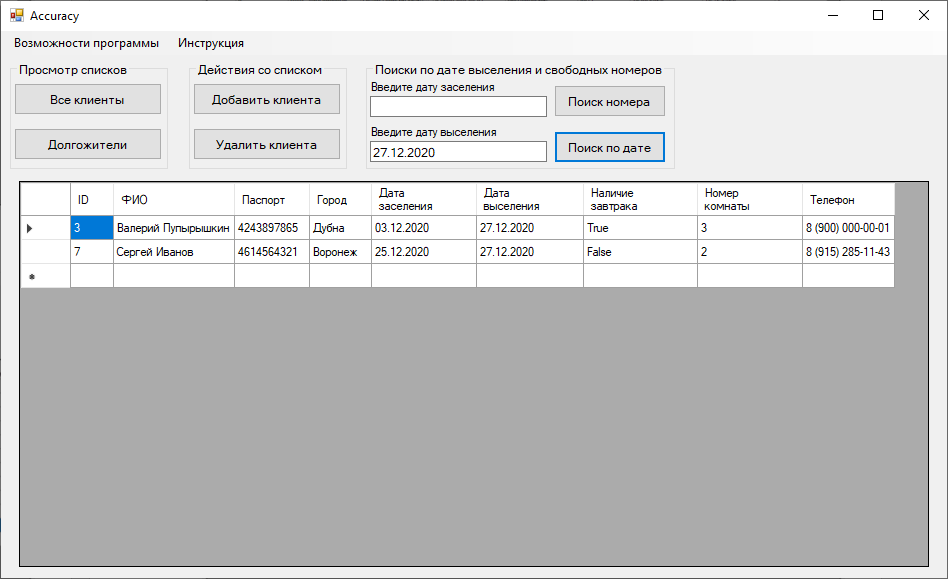


Рисунок 11 – Поиск клиентов по дате выселения

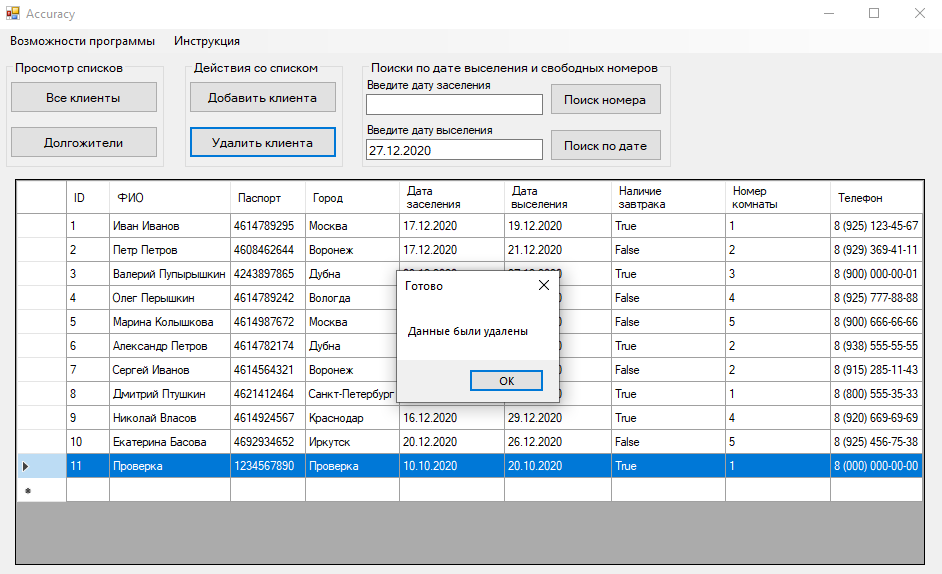


Рисунок 12 – Удаление данных о клиенте

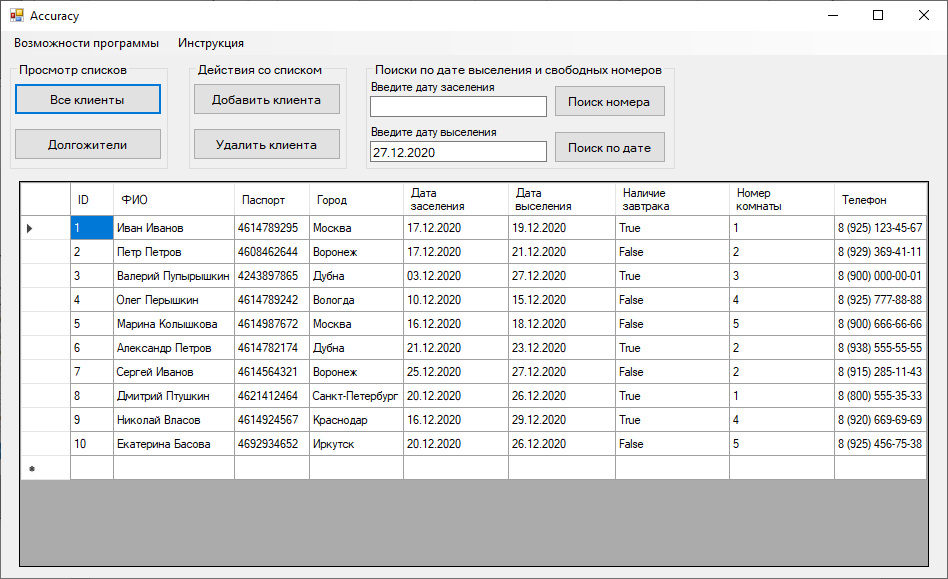


Рисунок 13 – Результат удаления данных о клиенте

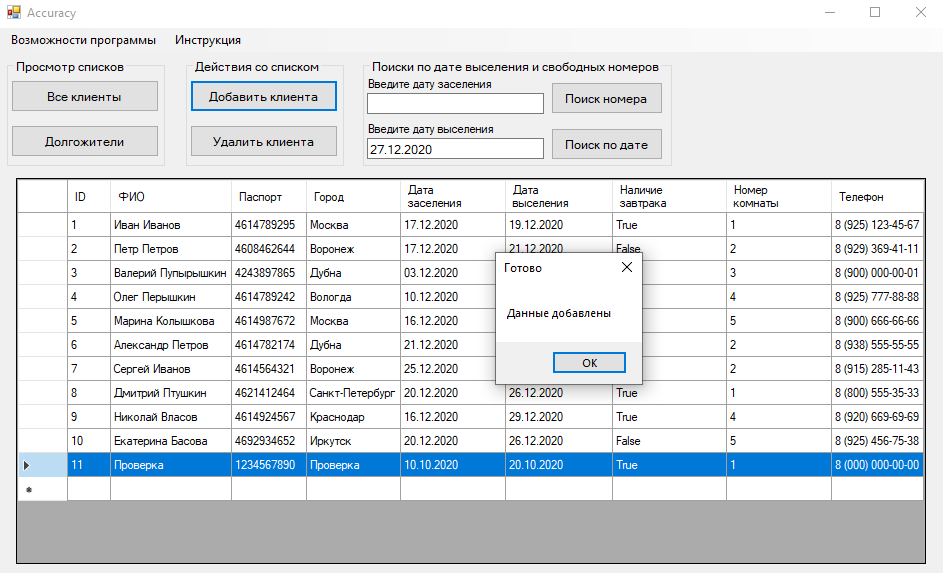


Рисунок 14 – Добавление данных о клиенте

# Руководство пользователя по работе с программной системой

## Установка системы

Установка системы производится распаковкой папки «Программа для установки» из архива. Запуск программы осуществляется при помощи файла «Accuracy.exe». Файл «hotel.mdb» является базой данных, в которую заносятся все данные, используемые программой.

## Работа в системе

Для облегчения работы с программой в саму систему встроено меню с инструкцией по её функциям – «Инструкция». Вид меню представлен на рисунке 15.

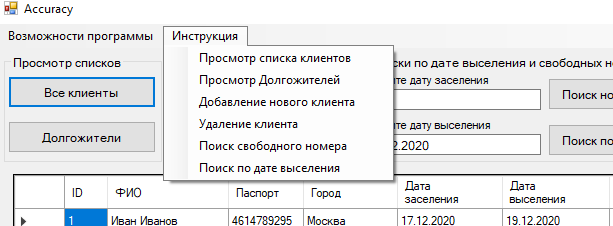


Рисунок 15 – Меню программы

Пункт меню «возможности программы» даёт представление о назначении программы и функциях, которые она реализовывает.

# Заключение

В результате написания курсовой работы была разработана Информационно-справочная система учета жителей гостиницы, что позволило закрепить полученные знания и научиться применять их на практике.

Разработанная система позволяет вести учет жителей гостиницы, внося новые данные и удаляя старые, и выводить информацию по запросам пользователей.

В ходе работы были выполнены поставленные задачи:

* Написано техническое задание;
* Разработаны схемы алгоритмов;
* Написана и отлажена информационно-справочная система «Информационно-справочная система учета жителей гостиницы».

# Список иллюстраций

[Рисунок 1 – Реализация структуры данных в СУДБ Microsoft Access 6](#_Toc59894967)

[Рисунок 2 – Общий алгоритм работы программы 7](#_Toc59894968)

[Рисунок 3 – Общий вид функции просмотра всех клиентов и долгожителей 7](#_Toc59894969)

[Рисунок 4 – Алгоритм добавления данных 8](#_Toc59894970)

[Рисунок 5 – Алгоритм удаления данных 9](#_Toc59894971)

[Рисунок 6 – Общий вид функций поиска клиентов по округу и сроку 9](#_Toc59894972)

[Рисунок 7 – Начальный экран программы 19](#_Toc59894973)

[Рисунок 8 – Просмотр всех клиентов гостиницы 19](#_Toc59894974)

[Рисунок 9 – Просмотр долгожителей гостиницы 20](#_Toc59894975)

[Рисунок 10 – Поиск свободных номеров 20](#_Toc59894976)

[Рисунок 11 – Поиск клиентов по дате выселения 21](#_Toc59894977)

[Рисунок 12 – Удаление данных о клиенте 21](#_Toc59894978)

[Рисунок 13 – Результат удаления данных о клиенте 22](#_Toc59894979)

[Рисунок 14 – Добавление данных о клиенте 22](#_Toc59894980)

[Рисунок 15 – Меню программы 23](#_Toc59894981)

# Список таблиц

[Таблица 1 – Структура данных 6](#_Toc59894982)