МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ

РОССИЙСКАЯ ФЕДЕРАЦИЯ

Федеральное государственное автономное образовательное учреждение

высшего образования

«ТЮМЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ИНСТИТУТ МАТЕМАТИКИ И КОМПЬЮТЕРНЫХ НАУК

КАФЕДРА ИНФОРМАЦИОННЫХ СИСТЕМ

КУРСОВАЯ РАБОТА

по дисциплине «Информатика и программирование»

«Разработка приложения для автоматизация учета продажи товаров»

Исполнитель:

студентка Исмакова Д.К.

группа 189-1

Руководитель:

канд. физ.- мат. наук,

доцент Моор П. К.

**СОДЕРЖАНИЕ**

[1. ПОСТАНОВКА ЗАДАЧИ 3](#_Toc10490695)

[2. ОПИСАНИЕ РАБОТЫ АЛГОРИТМОВ, КЛАССОВ, МЕТОДОВ 4](#_Toc10490696)

[2.1. Структура данных 4](#_Toc10490697)

[2.2. Описание работы «Главной формы» 5](#_Toc10490698)

[2.3. Описание работы формы «Клиенты» 5](#_Toc10490699)

[2.3.1. Описание работы метода LoadCl для считывания информации из файла. 6](#_Toc10490700)

[2.3.2. Описание работы метода DGV для записи данных в таблицу. 7](#_Toc10490701)

[2.3.3. Save\_btn\_Click\_1 (кнопка сохранения) 8](#_Toc10490702)

[2.3.4. Delete\_btn\_Click\_1 (кнопка удаления) 8](#_Toc10490703)

[2.3.5. New\_btn\_Click (создание нового элемента) 8](#_Toc10490704)

[2.3.6. Add\_btn\_Click (добавление нового элемента в таблицу) 8](#_Toc10490705)

[2.3.7. Change\_btn\_Click (изменение значений в таблице) 8](#_Toc10490706)

[2.4. Описание работы формы «Продажи» 9](#_Toc10490707)

[2.5. Описание работы формы «Продажи» 9](#_Toc10490708)

[2.6. Описание работы формы «Итоги» 10](#_Toc10490709)

[2.7. Описание работы формы «С подчиненной» 11](#_Toc10490710)

[3. ОПИСАНИЕ БАЗЫ ДАННЫХ ACCESS 13](#_Toc10490711)

[4. ЗАКЛЮЧЕНИЕ 15](#_Toc10490712)

# **ПОСТАНОВКА ЗАДАЧИ**

Основной целью данной работы является разработка Windows-приложения, реализующего функции СУБД такие, как добавление, удаление, редактирование данных, выборка. Приложение предназначено автоматизировать и облегчить ведение учета по продажам товаров.

Данное приложение будет работать с текстовыми файлами, содержащими информацию о товаре, клиенте и о самих продажах.

Основной функционал программы будет состоять из добавления новых записей в таблицы «Клиенты», «Товары», «Продажи» с сохранением, изменением и удалением записей; подведения итогов по всем продажам, а также выполнения выборки по коду клиента.

# ОПИСАНИЕ РАБОТЫ АЛГОРИТМОВ, КЛАССОВ, МЕТОДОВ

## Структура данных

Для начала определимся с используемыми переменными и способом хранения файлов. Названия основных переменных можно увидеть в таблицах, а все файлы будут храниться в текстовом формате(.txt).

Разделитель #

Файл клиенты client.txt

Файл client.txt будет хранить информацию с кодом клиента, его фамилией, а также с датой покупки товара.

Структура записи:

|  |  |
| --- | --- |
| **Код клиента** | **Фамилия** |
| code\_cl | fio |
| 1 | Лесив |
| 2 | Петров |

1#Иванов

Файл товары products.txt

Файл products.txt будет хранить информацию о товаре, цене. Также каждый товар будет иметь свой специальный код.

Структура записи:

|  |  |  |
| --- | --- | --- |
| **Код товара** | **Товар** | **Цена** |
| code\_prod | product | value |
| 1 | Ноутбук | 60 000 |
| 2 | Монитор | 25 000 |

1#Ноутбук#60000

Файл продажи sell.txt

Файл sell.txt будет хранить информацию с кодом продажи, внешними ключами от кода товара и кода клиента, с количеством, проданного товара и итог по проданным товарам

Структура записи:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Код продажи** | **Код клиента вн.кл.** | **Фамилия** | **Код товара вн.кл.** | **Товар** | **Цена** | **Кол-во** | **Дата покупки** |
| code\_sell | codeCl | fio | codeProd | product | value | numUpD | dt |
| 1 | 2 | Петров | 2 | Монитор | 25 000 | 2 | 12.03.2019 |

1#2#Петров#2#Монитор#25 000#2#12.03.2019#

## Описание работы «Главной формы»

На рисунке 1 представлен внешний вид Главной формы.

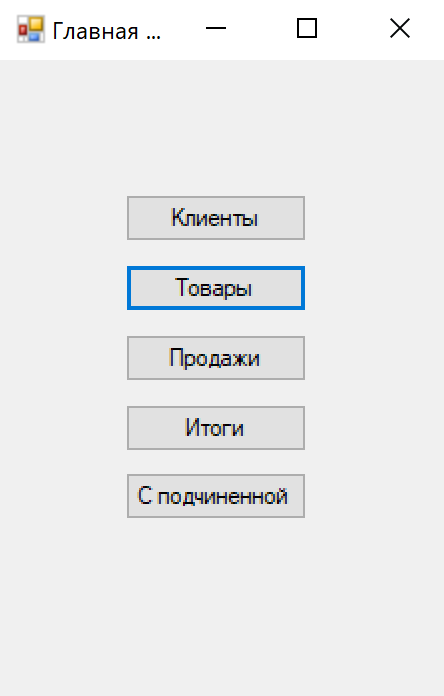


Рисунок 1 – Главная форма

На главной форме мы можем увидеть кнопки, которые переносят на одноимённые формы. Далее эти формы будут разобраны по отдельности.

## Описание работы формы «Клиенты»

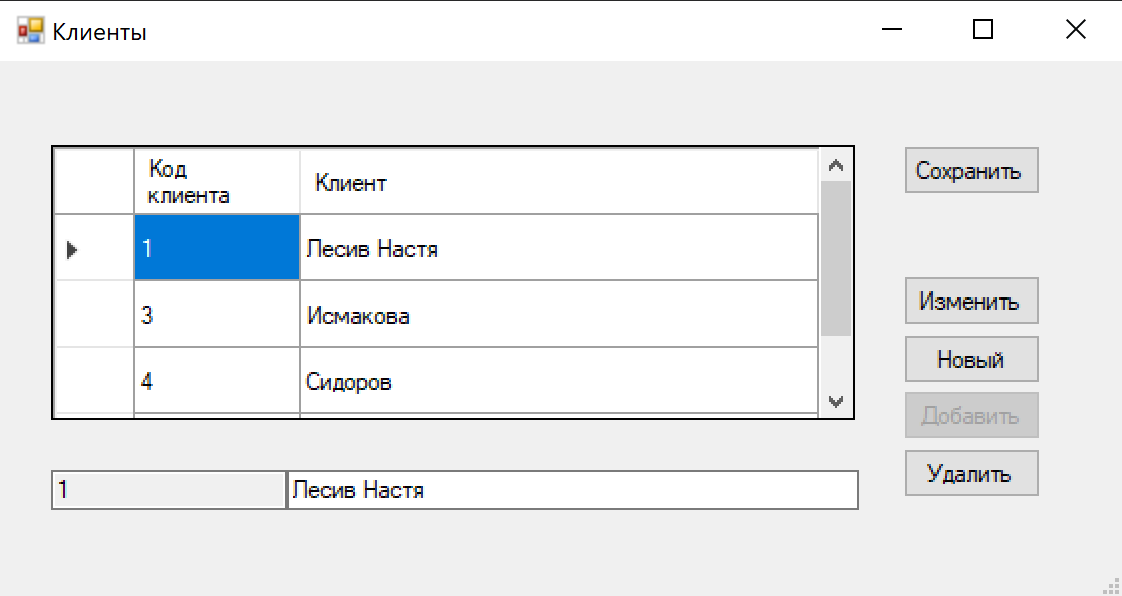


Рисунок 2 – форма «Клиенты»

На рисунке 2 представлена форма «Клиенты». Основная задача этой формы – вносить информацию о клиенте в таблицу с дальнейшей записью в файл.

При нажатии на клиента в области datagridview автоматически происходит синхронизация с вводимым полем. Если поменять данные, а потом нажать кнопку **Изменить**, то данные в таблице изменятся. Для того, чтобы добавить нового клиента, необходимо нажать кнопку **Новый**, тогда поле для ввода отчистится и изменится код клиента на новый. После заполнения нажимаем кнопку **Добавить,** и новый клиент вносится в таблицу. По завершению наживаем кнопку **Сохранить**, и все данные записываются в файл client.txt. Аналогично работает форма «Товары».

Теперь рассмотрим программный код.

### Описание работы метода LoadCl для считывания информации из файла.

Вот так выглядит код:

private void LoadCl()

{

try

{

string[] text = File.ReadAllLines("client.txt", Encoding.GetEncoding(1251));

HashSet<string> remEmpF = new HashSet<string>();

for (int i = 0; i < text.Length; i++)

{

remEmpF.Add(text[i]);

}

remEmpF.Where(x => !string.IsNullOrWhiteSpace(x));

string[] client\_txt = new string[0];

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

{

Array.Resize(ref client\_txt, client\_txt.Length + 1);

client\_txt[i] = remEmpF.ElementAt(i);

}

len = client\_txt.Length;

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

{

if (client\_txt[i] != "")

{

Array.Resize(ref code\_cl, len);

Array.Resize(ref fio, len);

string[] ss = client\_txt[i].Split(new char[] { '#' }, StringSplitOptions.RemoveEmptyEntries);

code\_cl[i] = int.Parse(ss[0]);

fio[i] = ss[1];

}

}

}

catch

{

MessageBox.Show("Данные в файле заполнены неправильно");

}

}

Загружаем данные из файла в массив text. Далее, чтобы избавиться от пустых строк используем переменную remEmpF, а уже хашсет(remEmpF) без пустых строк переписываем в массив client\_txt, с которым мы будем работать далее.

Для того, чтобы разделить массив client\_txt по нашему разделителю (#) необходимо сделать цикл, который будет увеличивать итерацию на один и идти по длине массива client\_txt, таким циклом будет цикл for.

Делаем проверку на пустоту массива client\_txt, если там ничего нет, цикл не запустится. В цикле мы делим строку client\_txt[i] на отдельные массивы, в одном будет храниться информация о коде клиента (code\_cl), а в другом его фамилия (fio).

Для отдельных случаев вылета программы предусмотрена конструкция try-catch, которая выводит окно с сообщением об ошибке.

### Описание работы метода DGV для записи данных в таблицу.

Вот так выглядит код:

private void DGV(int len)

{

dataGridView1.Rows.Clear();

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

{

dataGridView1.Rows.Add(code\_cl[i], fio[i]);

}

}

Вначале происходит полное очищение datagridview, чтобы информация не до записывалась, а полностью переписывалась (для избавления от одинаковой информации). Далее идет цикл for по необходимой длине len, добавляются наши массивы code\_cl и fio.

Далее пойдет код для кнопок в форме «Клиенты»

### Save\_btn\_Click\_1 (кнопка сохранения)

int countStr = dataGridView1.Rows.Count;

string[] saveClient = new string[countStr];

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

{

for (int j = 0; j < 2; j++)

{

saveClient[i] += dataGridView1.Rows[i].Cells[j].Value + "#";

File.WriteAllLines("client.txt", saveClient, Encoding.GetEncoding(1251));

}

}

MessageBox.Show("Данные сохранены");

### Delete\_btn\_Click\_1 (кнопка удаления)

dataGridView1.Rows.Remove(dataGridView1.Rows[num\_row]);

textBox1.Text = "";

textBox2.Text = "";

Array.Resize(ref code\_cl, len - 1);

Array.Resize(ref fio, len - 1);

len--;

### New\_btn\_Click (создание нового элемента)

add\_btn.Enabled = true;

if (dataGridView1.Rows[len - 1].Cells[1].Value != null)

{

code = code\_cl[len - 1];

len++;

Array.Resize(ref code\_cl, len);

Array.Resize(ref fio, len);

code\_cl[len - 1] = code + 1;

code++;

textBox1.Text = code.ToString();

textBox2.Text = "";

}

### Add\_btn\_Click (добавление нового элемента в таблицу)

add\_btn.Enabled = false;

if (textBox2.Text == "")

{

MessageBox.Show("Поле пустое");

}

else

{

fio[len - 1] = textBox2.Text;

DGV(len);

}

### Change\_btn\_Click (изменение значений в таблице)

fio[num\_row] = textBox2.Text;

DGV(len);

## Описание работы формы «Товары»

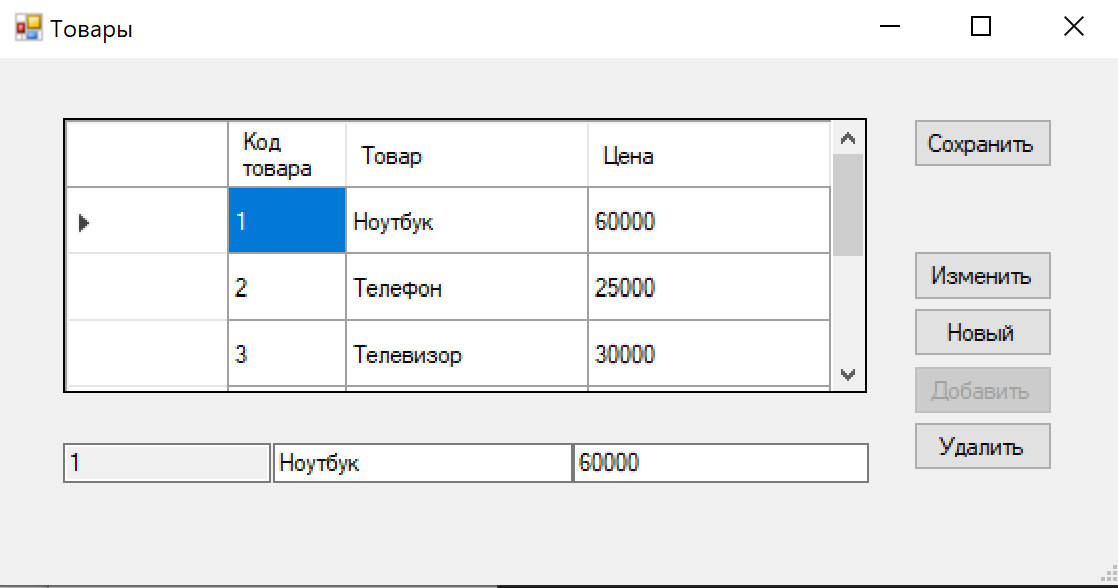


Рисунок 3 – форма «Товары»

На рисунке 3 представлен внешний вид формы «Продажи». Так как она работает аналогичным образом, как и форма клиенты, её описание можно опустить.

## Описание работы формы «Продажи»

Рисунок 4 – форма «Продажи»

Форма «Продажи» загружает в выпадающий список данные из файла с именами клиентов(client.txt), а также из файла с наименованиями товаров(product.txt). Помимо этого, заполняется таблица с информацией о предыдущих заказах из файла sell.txt, в который и будет записываться добавленная информация.

Данные в таблице синхронизированы с полем ввода данных. Все кнопки работают точно также, как и в форме «Клиенты».

## Описание работы формы «Итоги»

Рисунок 5 – форма «Итоги»

Итоги формируются по данным файлов client.txt (клиенты) и sell.txt (продажи). Основная выборка идет по коду клиента и далее подсчитывается сколько товаров он приобрел и на какую сумму. И таким образом проходит заполнение в таблицу.

Код для заполнения выборки в таблицу:

private void DGV()

{

dataGridView1.Rows.Clear();

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

{

int counter = 0;

int sumProd = 0;

int result=1;

dataGridView1.Rows.Add(code\_cl[i],fio[i]);

for(int j = 0; j < len1; j++)

{

if (dataGridView1.Rows[i].Cells[0].Value.ToString() == codeCl[j].ToString())

{

counter++;

sumProd += num[j];

result = sumProd \* value[j];

}

dataGridView1.Rows[i].Cells[2].Value = counter;

dataGridView1.Rows[i].Cells[3].Value = sumProd;

if (result == 1)

{

dataGridView1.Rows[i].Cells[4].Value = 0;

}

else

dataGridView1.Rows[i].Cells[4].Value = result;

}

}

}

## Описание работы формы «С подчиненной»

Рисунок 6 – форма «С подчиненной»

В форме «С подчиненной» имеется две таблицы. Вся информация заполняется из файлов client.txt (клиенты) и sell.txt (продажи). Первая таблица заполняется по фамилии клиента, а вторая, в соответствии с выбранным клиентом, по товарам, который этот клиент купил.

После считывания всей необходимой информации происходит заполнение таблиц.

Вот так выглядит код для записи во вторую таблицу, а также в поле «Итого»:

dataGridView2.Rows.Clear();

textBox1.Text = "";

int result = 0;

num\_row = dataGridView1.CurrentCell.RowIndex;

string kodeCl = dataGridView1.Rows[num\_row].Cells[0].Value.ToString();

for(int i = 0; i < len1; i++)

{

if (kodeCl == code\_client[i])

{

dataGridView2.Rows.Add(code\_sell[i], code\_prod[i], product[i], value[i], num[i], date[i]);

result += total[i];

textBox1.Text = result.ToString();

}

}

# ОПИСАНИЕ БАЗЫ ДАННЫХ ACCESS

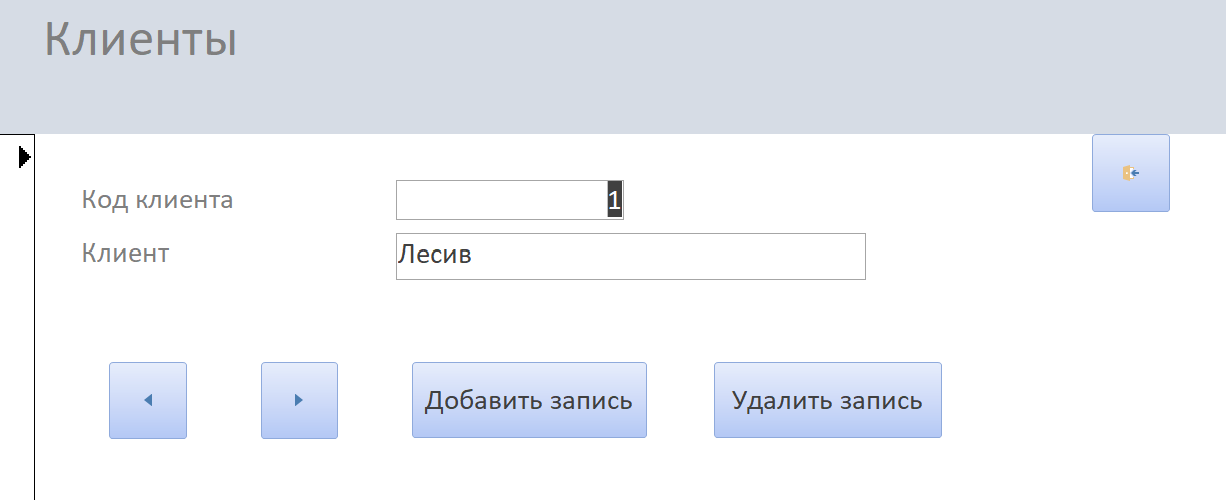
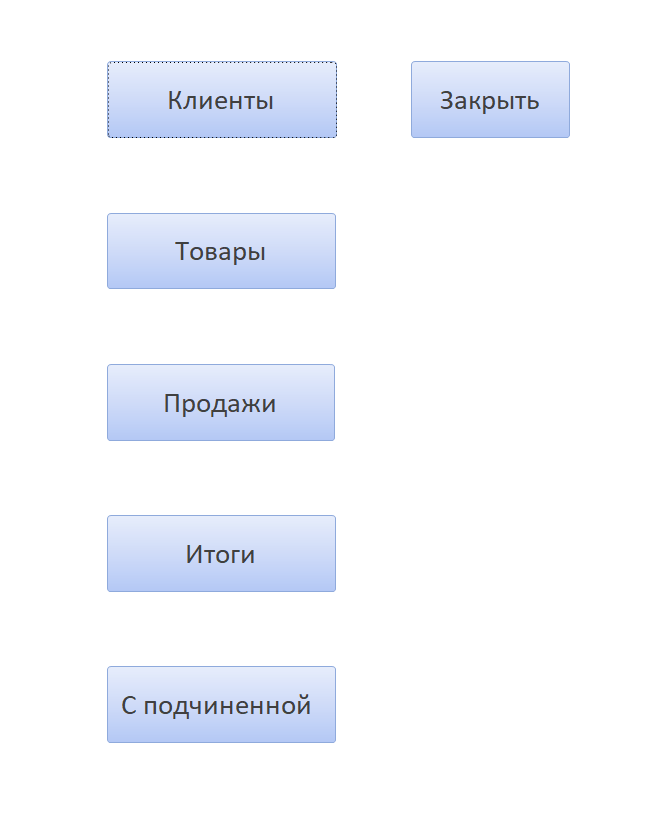
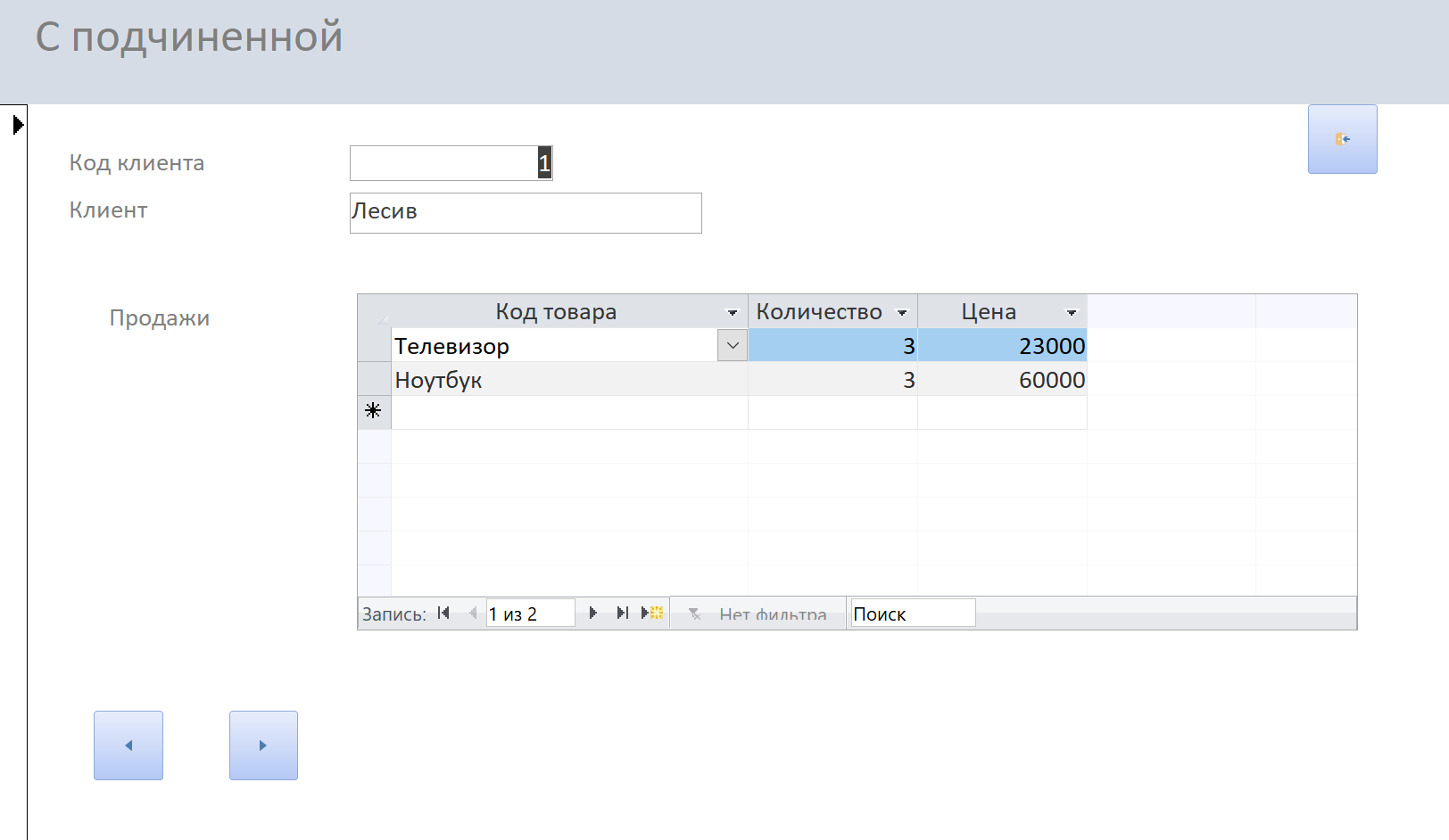
В базе данных имеется три основных таблицы: «Клиенты», «Товары» и «Продажи». На основе этих таблиц создаются формы с одноименными названиями. На главной форме находятся кнопки, ссылающие на другие формы: «Главная» (рис. 7), «Клиент» (рис. 8), «Товар», «Продажи», «С подчиненной» (рис. 9) и на отчет «Итоги» (рис. 10).

Рис. 9 – «С подчиненной»

Рис. 8 – «Клиенты»

Рис. 7 – «Главная»

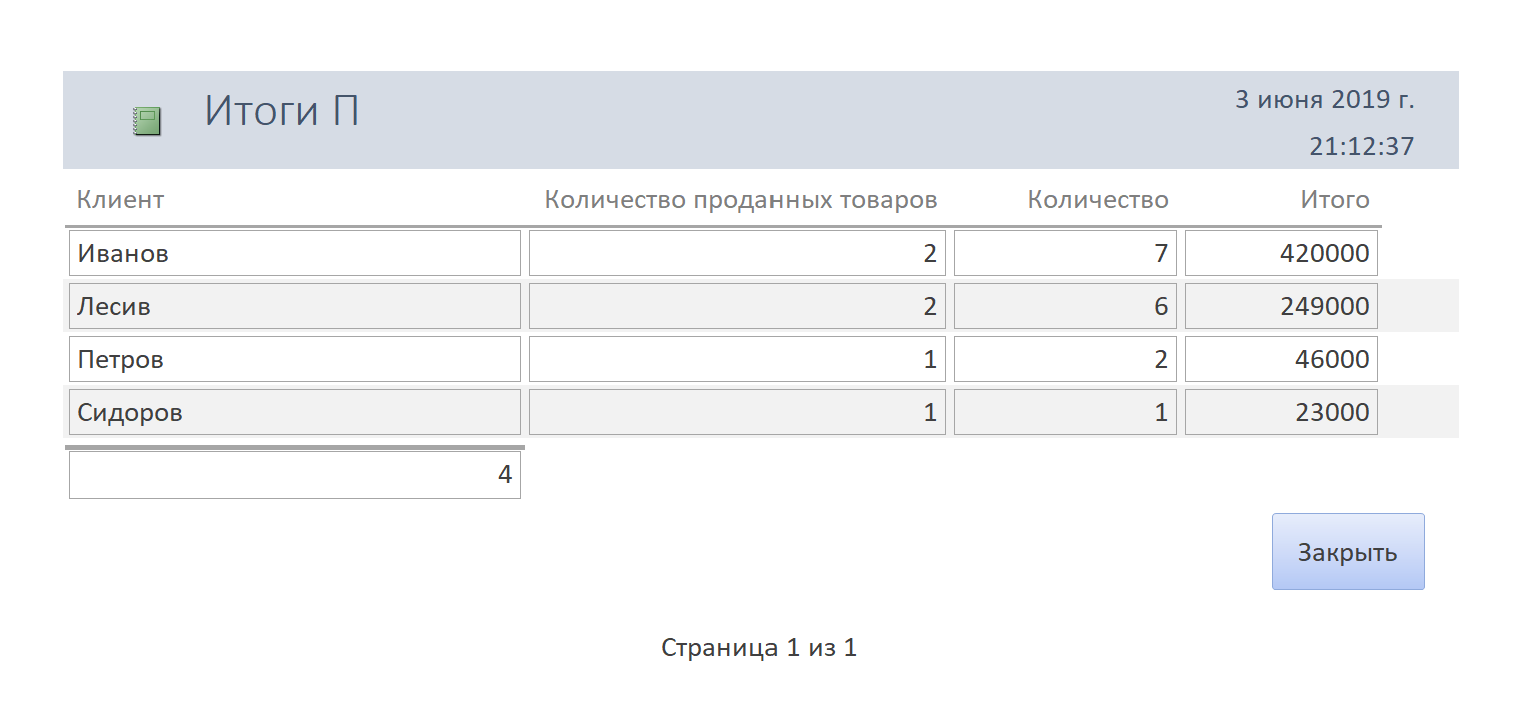
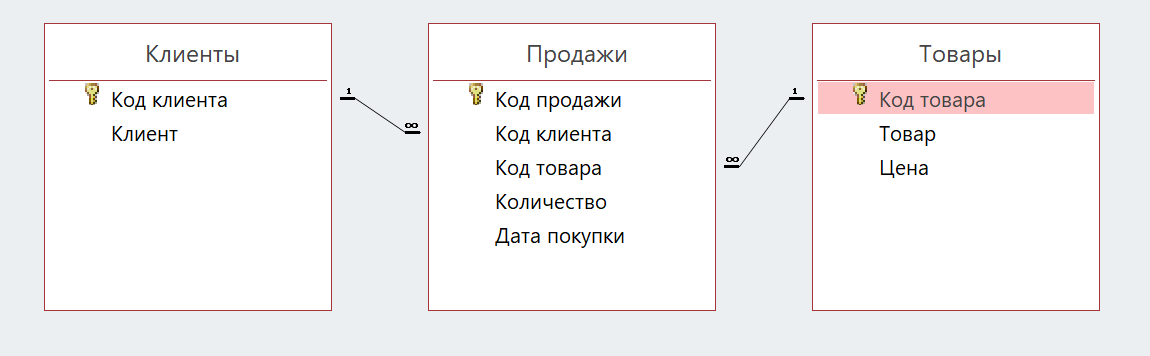
Функционал работы форм подобен Windows-формам, поэтому нет смысла описывать их работу заново. Так как это база данных, то основная составляющая работы – это схема данных, представленная на рисунке 11.

Рис. 10 – «Итоги»



Используется связь «один ко многим», которая предполагает возможность выбора нескольких данных из одной таблицы по коду в другой.

Рис. 11 – «Схема данных»

# ЗАКЛЮЧЕНИЕ

Во время выполнения курсовой работы было создано Windows-приложение позволяющее вести учет по продажам товаров. Оно может использоваться в различных сферах продажи товаров.

В ходе данной работы были выполнены следующие задачи:

* описаны алгоритмы, классы, методы, используемые в коде;
* описан интерфейс пользователя, элементов управления, предоставлены «снимки» экранных форм;
* описана база данных Access по учету продаж.

Основная цель курсовой работы по созданию Windows-приложения для учета продаж была выполнена.