МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ

РОССИЙСКОЙ ФЕДЕРАЦИИ

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 автоматизации учета выдачи книг в библиотеке “библиотека”»

Исполнитель:

Студент Бабешин К.А.

группа 185-2

Руководитель:

старший преподаватель

Павлова Е. А.

Тюмень – 2019

**Оглавление**

[Введение 3](#_Toc11804241)

[1.Постановка задачи 4](#_Toc11804242)

[2.Описание алгоритмов, классов и методов 5](#_Toc11804243)

[3.Описание программного приложения 23](#_Toc11804244)

[Заключение 27](#_Toc11804245)

[Литература 28](#_Toc11804246)

# Введение

В настоящее время зависимость людей от информационных систем увеличивается с каждой секундой. Человек уже не видит свою жизнь без компьютеров и программного обеспечения (далее ПО). Во многих организациях(компаниях) используется узконаправленное ПО.

Очень часто появляются различные компании, следовательно, растет спрос на создание ПО, которое будут работать с базой данных (далее БД). Для создания качественных информационных систем требуется небольшая группа программистов, которые будут знать несколько языков программирования и профессионально владеть ими. Поэтому спрос на хороших всегда большой.

# 1.Постановка задачи

Основной задачей данной курсовой работы было написать на языке программирования C# (Си шарп) программу для учета выдачи книг в библиотеке “библиотека”. В программном приложении требовалось осуществить хранение данных в текстовых файлах, это было реализовано в следующем виде:

Таблица -Книги

|  |  |  |  |
| --- | --- | --- | --- |
| Имя колонки | Тип колонки | Индексированное поле | Подпись |
| Column1 | Числовой | Да (ключ) | № паспорта книги |
| Column2 | Текстовый | Нет | Автор |
| Column3 | Текстовый | Нет | Название |
| Column4 | Дата | Нет | Дата издания |
| Column5 | Текст | Нет | издательство |

Таблица -Читатели

|  |  |  |  |
| --- | --- | --- | --- |
| Имя колонки | Тип колонки | Индексированное поле | Подпись |
| Column1 | Числовой | Да (ключ) | № паспорта читателя |
| Column2 | Текстовый | Нет | ФИО |
| Column3 | Текстовый | Нет | Адрес проживания |
| Column4 | Дата | Нет | Дата рождения |
| Column5 | Числовой | Нет | Утеряно книг |
| Column6 | Числовой | Нет | Номер телефона |

Таблица -Выдачи

|  |  |  |  |
| --- | --- | --- | --- |
| Имя колонки | Тип колонки | Индексированное поле | Подпись |
| Column1 | Числовой | Да (ключ) | № |
| Column2 | Текстовый | Нет | Название книги |
| Column3 | Текстовый | Нет | ФИО |
| Column4 | Дата | Нет | Дата выдачи |
| Column5 | Дата | Нет | Дата возврата |

# 2.Описание алгоритмов, классов и методов

Стоит заранее отметить, что большинство методов и алгоритмов в разработанном приложении между собой идентичны либо же имеют минимальные различия, потому подобные методы и алгоритмы будут представлены и описаны единожды, для экономия места и ресурсов.

Начинается всё с формы 1 (Form1), где при нажатии на каждую из кнопок вызывается новая форма (Код 1), и ей задается визуальные параметры родительской формы, благодаря следующему методу (для каждой кнопки создан свой метод, каждый из которых открывает разные формы):

Код 1

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

{

Form1 param = new Form1();

Form readers = new readersForm();

readers.ForeColor = ForeColor;

readers.BackColor = BackColor;

readers.Font = new Font(Font.Name, 8, FontStyle.Bold);

readers.ShowDialog();

}

Здесь создается переменная readers, отвечающая за действия с формой Читатели (readersForm), после для этой формы задаются параметры цвета фона, шрифта и стиль шрифта, только после этого форма открывается в как диалоговое окно, тем самым запрещая какие-либо манипуляции с родительской формой. Сделано это для того, чтобы не произошла ошибка при заполнении сразу нескольких таблиц.

Далее происходит переход к другой форме, например к форме 2 (Читатели):

Код 2

public string[] stringMassive;

public int numberCell, numberRow;

string[][] fulContent;

string[] numberPassport;

Перед инициализацией формы происходит объявление переменных (Код 2) stringMassive, numberCell, numberRow, fulContent, numberPassport, которые будут в дальнейшем связаны данными из файла readers.txt.

Далее происходит инициализация формы (Код 3), форма принимает внешний вид, который соответствует стилю, который выбрал пользователь или который задан по умолчанию. Затем вызываются основных метода OpenFile() и CrearteTablet(). Первый открывает файл readers.txt и записывает в массив stringMassive строки с разделителем “\n”, а затем удаляет пустые строки. Второй метод создает в элементе dataGridView, под названием readerDGV1, таблицу. Предварительно каждую строку из массива stringMassive, разбив строку по символу “@” и зависав данные в ступенчатый массив fulContent, для удобства проверки данных и обращению к ним создается массив numberPassport, в который записываются номера паспортов читателей (Код 4).

Код 3

private void v\_Load(object sender, EventArgs e){

OpenFile();

CrearteTablet();

readerDGV1.ForeColor = Color.Black;

save.BackColor = Color.Black;

save.ForeColor = Color.White;

edit.BackColor = Color.Black;

edit.ForeColor = Color.White;

del.BackColor = Color.Black;

del.ForeColor = Color.White;

add.BackColor = Color.Black;

add.ForeColor = Color.White;

search.BackColor = Color.Black;

search.ForeColor = Color.White;}

Код 4

private void OpenFile(){

string nameFile = @"readers.txt";

StreamReader f = new StreamReader(nameFile, Encoding.Default);

stringMassive = f.ReadToEnd().Split('\n');

f.Close();

stringMassive = stringMassive.Where(x => x != "").ToArray();}

private void CrearteTablet(){

readerDGV1.Rows.Clear();

numberPassport = new string[stringMassive.Length];

fulContent = new string[stringMassive.Length][];

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

fulContent[i] = stringMassive[i].Split('@');}

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

readerDGV1.Rows.Add();

for (int j = 0; j < 6; j++){

readerDGV1.Rows[i].Cells[j].Value = fulContent[i][j];

if (j == 0){

numberPassport[i] = Convert.ToString(int.Parse(readerDGV1.Rows[i].Cells[j].Value.ToString()));

}}}}

Затем описаны методы, которые вызываются при нажатии пользователем на кнопки, например кнопка Изменить:

private void edit\_Click(object sender, EventArgs e){

if (checkTextBoxs() && numberRow > -1){

readerDGV1.Rows[numberRow].Cells[0].Value = textBox1.Text;

readerDGV1.Rows[numberRow].Cells[1].Value = textBox2.Text;

readerDGV1.Rows[numberRow].Cells[2].Value = textBox3.Text;

readerDGV1.Rows[numberRow].Cells[3].Value = textBox4.Text;

readerDGV1.Rows[numberRow].Cells[4].Value = textBox5.Text;

readerDGV1.Rows[numberRow].Cells[5].Value = textBox6.Text;

stringMassive = new string[readerDGV1.RowCount];

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

for (int j = 0; j < 6; j++){

if (j != 5)

stringMassive[i] += readerDGV1.Rows[i].Cells[j].Value + "@";

else

stringMassive[i] += readerDGV1.Rows[i].Cells[j].Value;}}

CrearteTablet();

numberCell = 0;

numberRow = 0;

clearTextBoxs();}

else

MessageBox.Show("Строка не была выбрана!","ВНИМАНИЕ",MessageBoxButtons.OK);}

Здесь при нажатии на кнопку программа вызывает метод checkTextBoxs(), который проверяет введенные данные в текстовые поля на корректность (Код 5). Далее программа проверяет выбрана ли вообще строка, чтобы ее изменить. После данных проверок в выбранную строку таблицы readerDGV1 заполняются данные из текстовых полей , а сами поля отчищаются. Массив stringMassive инициализируется еще раз, и в него записываются новые данные из таблицы, склеивая ячейки каждой строки через символ “@” в строки массива. Затем снова вызывается метод CrearteTablet(), который перезаполняет таблицу readerDGV1, уже измененными данными.

Код 5

private bool checkTextBoxs(){

bool bl = true;

string date = @"^([0-9]{2}\/[0-9]{2}\/[0-9]{4})$";

string phone1 = @"^([0-9]\-[0-9]{2}\-[0-9]{2})$";

if (!int.TryParse(Convert.ToString(textBox1.Text).Trim(), out int aaa) || textBox1.Text.Length != 6){

MessageBox.Show("Поле [№ паспорта] было неверно запонено, проверьте правильность введенных данных", "", MessageBoxButtons.OK);

textBox1.Text = "";

bl = false;}

else if (textBox2.Text.Length == 0 || Convert.ToString(textBox2.Text).Trim() == "" || Convert.ToString(textBox2.Text).Trim() == " "){

MessageBox.Show("Поле [ФИО] было неверно запонено, проверьте правильность введенных данных", "", MessageBoxButtons.OK);

textBox2.Text = "";

bl = false;}

else if (textBox3.Text.Length == 0 || Convert.ToString(textBox3.Text).Trim() == "" || Convert.ToString(textBox3.Text).Trim() == " "){

MessageBox.Show("Поле [Адрес] было неверно запонено, проверьте правильность введенных данных", "", MessageBoxButtons.OK);

textBox3.Text = "";

bl = false;}

else if (!Regex.IsMatch(Convert.ToString(textBox4.Text).Trim(), date) || !DateTime.TryParse(Convert.ToString(textBox4.Text).Trim(), out DateTime dt1) ||

textBox4.Text.Length != 10){

MessageBox.Show(@"Поле [Дата рождения] было неверно запонено, проверьте правильность введенных данных.

Формат даты: дд/мм/гггг", "", MessageBoxButtons.OK);

textBox4.Text = "";

bl = false;}

else if (!int.TryParse(Convert.ToString(textBox5.Text).Trim(), out int baa) || textBox5.Text.Length == 0){

MessageBox.Show(@"Поле [Утеряно книг] было неверно запонено, проверьте правильность введенных данных.

строка должна иметь числовое значение", "", MessageBoxButtons.OK);

textBox5.Text = "";

bl = false;}

else if (textBox6.Text.Length != 0 &&( textBox6.Text.Length != 7 || Regex.IsMatch(Convert.ToString(textBox6.Text.ToString()).Trim(),phone1) == false)&& (textBox6.Text.Length != 11 || long.TryParse(Convert.ToString(long.Parse(textBox6.Text.ToString())).Trim(), out long phone) == false)){

MessageBox.Show(@"Поле [Номер телефона] было неверно запонено, проверьте правильность введенных данных.

строка должна иметь формат x-xx-xx или 11-значный номер", "", MessageBoxButtons.OK);

textBox6.Text = "";

bl = false;}

else if (int.TryParse(Convert.ToString(textBox1.Text).Trim(), out int a) && textBox1.Text.Length == 6 &&

textBox2.Text.Length != 0 && (Convert.ToString(textBox2.Text).Trim() != "" || Convert.ToString(textBox2.Text).Trim() != " ") &&

textBox3.Text.Length != 0 && (Convert.ToString(textBox3.Text).Trim() != "" || Convert.ToString(textBox3.Text).Trim() != " ") &&

Regex.IsMatch(Convert.ToString(textBox4.Text).Trim(), date) && DateTime.TryParse(Convert.ToString(textBox4.Text).Trim(), out DateTime dt) && textBox4.Text.Length == 10 &&

int.TryParse(Convert.ToString(textBox5.Text).Trim(), out int aa) && textBox5.Text.Length != 0 &&

(textBox6.Text.Length == 7 && Regex.IsMatch(Convert.ToString(textBox6.Text.ToString()).Trim(), phone1) || textBox6.Text.Length == 11 && long.TryParse(Convert.ToString(long.Parse(textBox6.Text.ToString())).Trim(), out long phoneone) || textBox6.Text.Length ==0))

bl = true; return bl;}

В случае, если пользователь заполнит текстовые поля и нажмет на кнопку Добавить запись, то вызовется метод, определенный для этой кнопки. Который сперва вызовет метод checkTextBoxs(), который проверит данные на корректность (Код 5), а затем проверит введенный паспорт со всеми другими паспортами в массиве numberPassport, если совпадений не будет то stringMassive увеличит количество элементов на 1 и последним элементом станет строка, склеенная из данных в текстовый полях через символ “@”. Снова вызовется метод CrearteTablet(), и таблица обновится новыми данными. Если же введенный паспорт уже существует, то программа скажет пользователю, что добавлять людей с одинаковым паспортом нельзя (Код 6).

Код 6

private void add\_Click(object sender, EventArgs e){

if (checkTextBoxs())

if (numberPassport.Contains(Convert.ToString(textBox1.Text).Trim()))

MessageBox.Show("Читатель с таким паспортом уже сусществует", "Ошибка", MessageBoxButtons.OK);

textBox1.Text = "";}

else{

Array.Resize(ref stringMassive, stringMassive.Length + 1);

string newLine = "";

newLine += Convert.ToString(textBox1.Text).Trim() + "@" + Convert.ToString(textBox2.Text).Trim() + "@" + Convert.ToString(textBox3.Text).Trim() + "@" +

Convert.ToString(textBox4.Text).Trim() + "@" + Convert.ToString(textBox5.Text).Trim() + "@" + Convert.ToString(textBox6.Text).Trim();

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

if (i == stringMassive.Length - 1)

stringMassive[i] = newLine;}

CrearteTablet();

numberCell = 0;

numberRow = 0;

clearTextBoxs();}}

При нажатии на кнопку Удалить происходят следующее: программа меняет строку массива stringMassive с индексом, равным выбранным индексу выбранной строки таблицы readerDGV1, и заменит её на пустую строку, а затем удалит пустую строку из массива встроенным методом Where(). Снова перезапишет таблицу новыми данными (Код 7).

Код 7

private void del\_Click(object sender, EventArgs e){

if (numberRow > 0){

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

if (i == numberRow)

stringMassive[i] = "";

else

stringMassive[i] = stringMassive[i];}

stringMassive = stringMassive.Where(x => x != "").ToArray();

CrearteTablet();

numberCell = 0;

numberRow = 0;

clearTextBoxs();}}

При нажатии пользователем на кнопку Сохранить программа проверит не пустой ли массив stringMassive, если он пустой, то программа предупредит пользователя что он пытается сохранить пустую таблицу(это означает, что пользователь удалил все записи), если же массив не пустой, то метод, предопределенный кнопке откроет файл readers.txt и перезапишет в него данные из массива stringMassive с разделителем “\n” (Код 8).

Код 8

private void save\_Click(object sender, EventArgs e){

if (stringMassive.Length > 0){

StreamWriter f = new StreamWriter(@"readers.txt", false, System.Text.Encoding.Default);

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

if (i != stringMassive.Length - 1)

f.WriteLine(stringMassive[i]);

else

f.Write(stringMassive[i]);}

f.Close();

MessageBox.Show("Таблица успешно сохранена", "");}

else{

DialogResult dialogResult = MessageBox.Show("Вы хотите сохранить пустую таблицу, после сохранения все данные будут потеряный!", "ВНИМАНИЕ", MessageBoxButtons.YesNo);

if (dialogResult == DialogResult.Yes){

StreamWriter f = new StreamWriter(@"readers.txt", false, System.Text.Encoding.Default);

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

if (i != stringMassive.Length - 1)

f.WriteLine(stringMassive[i]);

else

f.Write(stringMassive[i]); }

f.Close();MessageBox.Show("Таблица успешно сохранена", "");}}}

При заполнении текстового поля в рамке Поиск кнопка Поиск станет активной и ее можно будет нажать. При нажатии на данную кнопку программа проверит является ли введенная строка номером паспорта, и выделит строку с введенным паспортом, тем самым предоставив данную строку для удаления или редактирования (Код 9).

Код 9

private void search\_Click(object sender, EventArgs e){

bool search = false;

string passport = searchTBX.Text.ToString();

for (int i = 0; i < readerDGV1.Rows.Count; i++){

if (passport == Convert.ToString(int.Parse(readerDGV1[0, i].Value.ToString()))){

search = true;

readerDGV1.CurrentCell = readerDGV1.Rows[i].Cells[0];

numberRow = readerDGV1.CurrentCell.RowIndex;

rowNumber.Text = Convert.ToString(numberRow + 1);

textBox1.Text = Convert.ToString(readerDGV1.Rows[numberRow].Cells[0].Value);

textBox2.Text = Convert.ToString(readerDGV1.Rows[numberRow].Cells[1].Value);

textBox3.Text = Convert.ToString(readerDGV1.Rows[numberRow].Cells[2].Value);

textBox4.Text = Convert.ToString(readerDGV1.Rows[numberRow].Cells[3].Value);

textBox5.Text = Convert.ToString(readerDGV1.Rows[numberRow].Cells[4].Value);

textBox6.Text = Convert.ToString(readerDGV1.Rows[numberRow].Cells[5].Value);}

}if (!search){

MessageBox.Show("Записи с таким паспортом не найдено", "Ошибка");}}

При закрытии формы вызовется событие закрытия формы. Программа предложит пользователю сохранить данные таблицы, если пользователь в диалоговом окне dialogResult нажмет на кнопку OK, то метод откроет файл

readers.txt и перезапишет в него данные из массива stringMassive с разделителем “\n” (Код 8).

Все представленные выше методы практически идентичны в формах Читатели (readersForm), Книги (booksForm)\_ и Выдачи (Output). Отличаются они количеством текстовых полей в формах, которые проверяются на корректные данные методом checkTextBoxs() (Код 5), также отличие в самих текстовых полях. В форме Выдачи (Output) первые два текстовых поля, отвечающие за название книги и ФИО читателя, являются элементом comboBox, заполнить которые можно как названиями и ФИО, так и № паспортов книг и читателей соответственно.

Код 10

private void itog\_Click(object sender, EventArgs e){

Form itog = new itog();

itog.ForeColor = ForeColor;

itog.BackColor = BackColor;

itog.Font = new Font(Font.Name,8, FontStyle.Bold);

itog.ShowDialog();}

private void subitog\_Click(object sender, EventArgs e){

Form subtab = new Subtabs();

subtab.ForeColor = ForeColor;

subtab.BackColor = BackColor;

subtab.Font = new Font(Font.Name,8, FontStyle.Bold);

subtab.ShowDialog();}

Также из формы 1 (Form1) при нажатии на кнопки Итоги (itog) и Итоги с подчинённой (subitog) вызываются формы itog и subitog соответственно (Код 10).

Код 11

public int numberCell, numberRow;

string[] booksMassive, readersMassive, outputMassive;

string[][] bookContent, readersContent, outputContent;

int bookEmpty, dateLose;

List<int> index\_book\_date\_lose;

int[] indexEmptyBooks, dateLoseBooks;

private void itog\_Load(object sender, EventArgs e){

OpenFile();

CrearteTablet();

allEmptyBook.Text = "";

label5.Text = "";

returnTab.Visible = false;

itogDGV1.ForeColor = Color.Black;

returnTab.BackColor = Color.Black;

returnTab.ForeColor = Color.White;

showEmptyBooks.BackColor = Color.Black;

showEmptyBooks.ForeColor = Color.White;

label1.BackColor = Color.White;

label2.BackColor = Color.White;

label3.BackColor = Color.White;

label1.ForeColor = Color.Black;

label2.ForeColor = Color.Black;

label3.ForeColor = Color.Black;

label4.BackColor = Color.White;

label4.ForeColor = Color.Black;

label5.BackColor = Color.White;

label5.ForeColor = Color.Black;

allLines.BackColor = Color.White;

allEmptyBook.BackColor = Color.White;}

При открытии формы Итоги, происходит тоже самое, что и в предыдущих формах. Сначала инициализируются переменные public int numberCell, numberRow;string[] booksMassive, readersMassive, outputMassive;

string[][] bookContent, readersContent, outputContent; int bookEmpty, dateLose;

List<int> index\_book\_date\_lose, в которых дальше будут храниться данные из трех файлов с данными вызываются методы OpenFile() и CrearteTablet() ,затем для элементов формы задаются визуальные параметры (Код 11). Как видно из названия методов, можно сказать, что они выполняют те же функции что и методы с аналогичными названиями их вышеописанных форм (Код 4), но здесь имеются кардинальные отличия. Начнем с метода OpenFile() (Код 12).

Код 12

private void OpenFile(){

StreamReader outputs = new StreamReader(@"output.txt", Encoding.Default);

outputMassive = outputs.ReadToEnd().Split('\n');

outputs.Close();

outputMassive = outputMassive.Where(x => x != "").ToArray();

StreamReader books = new StreamReader(@"books.txt", Encoding.Default);

booksMassive = books.ReadToEnd().Split('\n');

books.Close();

StreamReader readers = new StreamReader(@"readers.txt", Encoding.Default);

readersMassive = readers.ReadToEnd().Split('\n');

readers.Close();}

Метод OpenFile() открывает три файла output.txt, books.txt, readers.txt, записывает строки из них в массивы outputMassive (выдачи) , booksMassive (книги), readersMassive (читатели) с разделителем “\n” соответственно.

Метод CrearteTablet() также имеет отличия от метода с этим же названием в вышеперечисленных формах.

Сначала метод инициализирует глобальные массивы bookContent, readersContent, outputContent. Затем в них записывает данные с разделителем “@” из массивов, booksMassive (книги), readersMassive (читатели) соответственно. В массив outputContent при каждой итерации метод записывает данные из outputMassive с разделителем “@” и добавляя в таблицу itogDGV1 новую пустую строку, заполняет сперва столбцы Название книги и автора в соответствие с № паспорта из массива с данными о выдачах (outputContent), затем заполняя столбцы заполнение ФИО и адрес проживания в соответствии с № паспорта из массива с данными о выдачах (outputContent), затем заполняются столбцы с датами выдачи и возврата из массива outputContent, далее в этой же итерации метод проверяет данную запись на разность текущей даты и даты выдачи, дата возврата не заполнена и разность дат превышает 15 дней, то индекс данной записи добавляется в массив dateLoseBooks, но если разница дат не превышает 15 дней и дата возврата не заполнена, то индекс записи добавляется в массив indexEmptyBooks. На этом действия в итерации кончаются. После прохода цикла for всех итераций равных количеству строк в массиве outputMassive в текст элементов label с именами allLines, allEmptyBook и label5 записываются количество записей, количество несданных книг и количество просроченных книг соответственно (Код 13).

Код 13

private void CrearteTablet(){

index\_book\_date\_lose = new List<int>();

bookEmpty = 0;

dateLose = 0;

indexEmptyBooks = new int[0];

dateLoseBooks = new int[0];

//инициализация массивов со всеми данными

//из строк, считанных из файлов

bookContent = new string[booksMassive.Length][];

readersContent = new string[readersMassive.Length][];

outputContent = new string[outputMassive.Length][];

//заполнение массивов данными из строк с разделителем "@"

//КНИГИ

for (int i = 0; i < bookContent.GetLength(0); i++){

bookContent[i] = booksMassive[i].Split('@');

Convert.ToString(bookContent[i].ToString());}

//ЧИТАТЕЛИ

for (int i = 0; i < readersContent.GetLength(0); i++){

readersContent[i] = readersMassive[i].Split('@');

Convert.ToString(readersContent[i].ToString());}

//Создание таблицы

itogDGV1.Rows.Clear();

//ВЫДАЧИ

for (int i = 0; i < outputContent.GetLength(0); i++){

outputContent[i] = outputMassive[i].Split('@');

itogDGV1.Rows.Add();

for (int j = 0; j < bookContent.GetLength(0); j++){

if (bookContent[j][0] == outputContent[i][0]) {//заполняюся полей автор и название книги в соответствии с № паспорта в выдачах

itogDGV1.Rows[i].Cells[0].Value = bookContent[j][1];

itogDGV1.Rows[i].Cells[1].Value = bookContent[j][2]; }}

for (int j = 0; j < readersContent.GetLength(0); j++){

if (readersContent[j][0] == outputContent[i][1]){//заполнение Фио и адрес проживания в соответствии с № паспорта в выдачах

itogDGV1.Rows[i].Cells[2].Value = readersContent[j][1];

itogDGV1.Rows[i].Cells[3].Value = readersContent[j][2];

}}itogDGV1.Rows[i].Cells[4].Value = outputContent[i][2];//заполнение даты выдачи

itogDGV1.Rows[i].Cells[5].Value = outputContent[i][3];//заполнение даты возврата

if (outputContent[i][3].Length < 10 || Convert.ToString(outputContent[i][3].ToString()) == ""){//посчет несданных книг

bookEmpty++;

Array.Resize(ref indexEmptyBooks, indexEmptyBooks.Length + 1);

indexEmptyBooks[indexEmptyBooks.Length - 1] = i;}

if ((outputContent[i][3].Length < 10 || Convert.ToString(outputContent[i][3].ToString()) == "") && (DateTime.Now - Convert.ToDateTime(outputContent[i][2].ToString())).TotalDays > 15) {//подсчет просроченных книг

dateLose++;

Array.Resize(ref dateLoseBooks, dateLoseBooks.Length + 1);

dateLoseBooks[dateLoseBooks.Length - 1] = i;

index\_book\_date\_lose.Add(i); }}

allLines.Text = Convert.ToString(outputContent.GetLength(0));

allEmptyBook.Text = Convert.ToString(bookEmpty);

label5.Text = Convert.ToString(dateLose);}

Напротив записей о количестве несданных и просроченных книг находятся кнопки Показать (showEmptyBooks) и (show2) соответственно. При нажатии на одну из них вызовется метод соответствующий каждой кнопке и перезапишет таблицу в соответствии с подписью. В первом случае таблица itogDGV1 будет заполнятся лишь теми записями, чей индекс равен элементу из массива indexEmptyBooks, т.е. несданными книгами, во втором случае таблица itogDGV1 будет заполнятся лишь теми записями, чей индекс равен элементу из массива dateLoseBooks, т.е. просроченными книгами, плюс разблокируется кнопка Создать квитанции (CreteDoc). При нажатии на кнопку Создать квитанции (CreteDoc) вызовется соответствующий метод, будет создана переменная wordApp, которая будет отвечать за рабочую среду приложения Microsoft Word. Затем программа откроет заранее созданный файл-шаблон mulct.dot и впишет в него ФИО читателя, рассчитает штраф за просроченные дни, файл будет создан отдельно для каждой записи (Код 14).

Код 14

private void CreteDoc\_Click(object sender, EventArgs e){try{

for (int i = 0; i < dateLose; i++){

string name1 = itogDGV1[2, i].Value.ToString();

string date = DateTime.Now.ToShortDateString().ToString();

double money = Math.Round((Convert.ToDateTime(DateTime.Now.ToShortDateString()) - Convert.ToDateTime(outputContent[index\_book\_date\_lose[i]][2].ToString())).TotalDays - 15, 2); var wordApp = new Word.Application();

wordApp.Visible = false;

var wordDocument = wordApp.Documents.Add(@"" + Application.StartupPath.ToString() + @"\mulct.dot");

RepalceWord("{name}", name1, wordDocument);

RepalceWord("{name}", name1, wordDocument);

RepalceWord("{book}", itogDGV1[1, i].Value.ToString(), wordDocument);

RepalceWord("{money}", money.ToString(), wordDocument);

RepalceWord("{date}", date, wordDocument);

wordApp.Visible = true;}}

catch (Exception){throw;}}

private void RepalceWord(string stubToReplace, string text, Word.Document wordDocument){var range = wordDocument.Content;

range.Find.ClearFormatting();

range.Find.Execute(FindText: stubToReplace, ReplaceWith: text);}

При нажатии на кнопку Форма Итоги с подчинённой (subitog) ,как ясно из названия является формой с подчиненной, реализованной через взаимодействие comboBox и dataGridView, в первом выбирается элемент, а во втором отображаются все связанные с ним записи в выдаче. При изменении значения в ComboBox таблица subtab отчищается и заполняется данными из других баз данных (далее БД). Сравнивается № паспорта выбранного читателя с № паспорта с № паспорта в БД Выдачи. В таблицу subtab записываются все книги, которые брал выбранный читатель. Также выводится в label’ы количество взятых, несданных и просроченных книг. Если у читателя есть просроченные книги активируется кнопка Создать квитанцию (createdoc), при нажатии на которую также как и в предыдущей форме открывается файл-шаблон mulct.dot и впишет в него ФИО читателя, рассчитает штраф за просроченные дни, файл будет создан отдельно для каждой записи (Код 15).

Код 15

private void tabCreate(){

index\_book\_date\_lose = new List<int>();

booklose = 0;

emptyDatetwo = 0;

string pasportNumber = "";

subtab.Rows.Clear();

for (int i = 0; i < readersContent.GetLength(0); i++){

if (readersList.Items[readersList.SelectedIndex].ToString() == readersContent[i][1]){pasportNumber = readersContent[i][0];}}

pasportNumb = new string[0];

for (int i = 0; i < outputContent.GetLength(0); i++){if (pasportNumber == outputContent[i][1]){

Array.Resize(ref pasportNumb, pasportNumb.Length + 1);

pasportNumb[pasportNumb.Length - 1] = outputMassive[i];

if ((outputContent[i][3].Length < 10 || outputContent[i][3].ToString() == "") && (DateTime.Now - Convert.ToDateTime(outputContent[i][2].ToString())).TotalDays > 15){

emptyDatetwo++;

index\_book\_date\_lose.Add(i);}

if (outputContent[i][3].Length < 10 || outputContent[i][3].ToString() == "") booklose++;}}

string[][] outputContentTwo = new string[pasportNumb.Length][];

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

outputContentTwo[i] = pasportNumb[i].Split('@');}

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

subtab.Rows.Add();

for (int j = 0; j < bookContent.GetLength(0); j++){

if (bookContent[j][0] == outputContentTwo[i][0]){

subtab.Rows[i].Cells[0].Value = bookContent[j][2];

subtab.Rows[i].Cells[1].Value = bookContent[j][1];}}

subtab.Rows[i].Cells[2].Value = outputContentTwo[i][2];

subtab.Rows[i].Cells[3].Value = outputContentTwo[i][3];}}

Форма Запросы (request) представляет собой набор кнопок, при нажатии на которые в таблицу mainGrid выводятся запросы, соответствующие кнопке. Так можно просмотреть информацию о взятых книгах за определенный месяц-год, самые популярные книги, самые не популярные книги, узнать самый популярный месяц, в который берут книги, и самого любимого читателя, который взял больше всех книг. Первые три запроса организуются из одного метода Select\_Item(int exclus), в данный метод передается значение, которые определяет ищем ли самую популярную книгу или наоборот. Сперва в словарь countTable записываются номера месяцев или паспорт книги как ключ, а значение количество записей с данными ключами в БД Выдачи и заполняется таблица mainGrid книгами, у которых самое большое и самое маленькое значение (Код 16). Для четвертого запроса вызывается метод search\_Month(), который в словарь countTable\_for\_mounth записываются номера месяцев как ключ, а в значения записывается количество записей в данный месяц. Затем заполняется таблица mainGrid книгами, взятыми за месяц с самым большим значением (Код 17). Для последнего запроса вызывается метод search\_Reader(), который в словарь countTable\_for\_readers записывает № паспорта читателя как ключ, а как значение записывается количество взятых книг. В таблицу mainGrid выводятся все читатели, которые брали больше всех книг (Код 18).

Код 16

private void Select\_Item(int exclus){

mainGrid.Columns.Clear();

countTable = new Dictionary<string, int>();

for (int i = 0; i < 4; i++)

mainGrid.Columns.Add("column\_" + (i + 1), clmnName[i]);

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

if (Convert.ToString(dateContent[i][2].ToString()) == selectYear && selectMonth == Convert.ToString(dateContent[i][1].ToString()) && selectMonth != "0" && selectYear != "0")//если совпадают даты выдачи книг{

int count = 0;

for (int j = 0; j < booksMassive.Length; j++){

if (Convert.ToString(bookContent[j][0].ToString()) == Convert.ToString(outputContent[i][0].ToString())){

count++;

//если ключ (книга) есть в словаре то счетчик увеличивается

//в противном случае создается новая запись

if (!countTable.ContainsKey(Convert.ToString(outputContent[i][0].ToString()))) countTable.Add(Convert.ToString(outputContent[i][0].ToString()), count);

else countTable[Convert.ToString(outputContent[i][0].ToString())] += 1;}}}

else if (selectMonth == "0" && selectYear == "0"){

int count = 0;

for (int j = 0; j < booksMassive.Length; j++){

if (Convert.ToString(bookContent[j][0].ToString()) == Convert.ToString(outputContent[i][0].ToString())){

count++;

//если ключ (книга) есть в словаре то счетчик увеличивается

//в противном случае создается новая запись

if (!countTable.ContainsKey(Convert.ToString(outputContent[i][0].ToString()))) countTable.Add(Convert.ToString(outputContent[i][0].ToString()), count);

else countTable[Convert.ToString(outputContent[i][0].ToString())] += 1;}}}}

//сортировка по значениям

var myList = countTable.ToList();

if (exclus == 0)//самая популярная

myList.Sort((pair1, pair2) => pair2.Value.CompareTo(pair1.Value));

else//самая не популярная

myList.Sort((pair1, pair2) => pair1.Value.CompareTo(pair2.Value));

newList\_2 = new Dictionary<string, int>();

try{

string pass = myList.First().Key.ToString();

int countPass = myList.First().Value;

foreach (var el in countTable)

if (el.Value == countPass)

newList\_2.Add(el.Key, countTable[el.Key]);

mainGrid.Rows.Clear();

CrearteTablet(pass, 0);

if (selectMonth == "0" && selectYear == "0")

label3.Text = "Количество читателей взявших данную книгу : " + myList.First().Value.ToString();

else

label3.Text = "Количество книг взятых за данный период : " + myList.First().Value.ToString();}

catch{MessageBox.Show("В данный период книги не брались");}}

Код 17

private void search\_Month(){

mainGrid.Columns.Clear();

countTable\_for\_mounth = new Dictionary<string, int>();

for (int i = 0; i < 4; i++)

mainGrid.Columns.Add("column\_" + (i + 1), clmnName[i]);

for (int i = 0; i < dateContent.GetLength(0); i++){

int count = 0;

count++;

//если ключ (книга) есть в словаре то счетчик увеличивается

//в противном случае создается новая запись

if (!countTable\_for\_mounth.ContainsKey(Convert.ToString(dateContent[i][1].ToString()))) countTable\_for\_mounth.Add(Convert.ToString(dateContent[i][1].ToString()), count);

else countTable\_for\_mounth[Convert.ToString(dateContent[i][1].ToString())] += 1;}

var myList = countTable\_for\_mounth.ToList();

myList.Sort((pair1, pair2) => pair2.Value.CompareTo(pair1.Value));

try{

string mounth = myList.First().Key.ToString();

mainGrid.Rows.Clear();

CrearteTablet(mounth, 2);

label3.Text = "Самый популярный месяц : " + mounthName[mounth].ToString();}

catch {MessageBox.Show("В данный период книги не брались");}}

Код 18

private void search\_Reader(){

mainGrid.Columns.Clear();

countTable\_for\_readers = new Dictionary<string, int>();

for (int i = 0; i < 4; i++)

{

mainGrid.Columns.Add("column\_" + (i + 1), clmnName\_for\_readers[i]);

}

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

int count = 0;

for (int j = 0; j < readersMassive.Length; j++){

if (Convert.ToString(readersContent[j][0].ToString()) == Convert.ToString(outputContent[i][1].ToString())){

count++;

//если ключ (книга) есть в словаре то счетчик увеличивается

//в противном случае создается новая запись

if (!countTable\_for\_readers.ContainsKey(Convert.ToString(outputContent[i][1].ToString()))) countTable\_for\_readers.Add(Convert.ToString(outputContent[i][1].ToString()), count);

else countTable\_for\_readers[Convert.ToString(outputContent[i][1].ToString())] += 1;

}}}

var myList = countTable\_for\_readers.ToList();

myList.Sort((pair1, pair2) => pair2.Value.CompareTo(pair1.Value));

newLiist = new Dictionary<string, int>();

string pass = myList.First().Key.ToString();

int countPass = myList.First().Value;

foreach (var el in countTable\_for\_readers)

if (el.Value == countPass)

newLiist.Add(el.Key.ToString(), countTable\_for\_readers[el.Key]);

mainGrid.Rows.Clear();

CrearteTablet(pass, 1);

label3.Text = "Количество взятых книг : " + myList.First().Value.ToString(); }

# 3.Описание программного приложения

При запуске приложения перед пользователем появляется главная форма, являющаяся меню, из которого он может перейти к формам для редактирования, добавления или удаления записей, форме с подчиненной или форме со статистическими данными.

Изображение выглядит как снимок экрана

Описание создано с очень высокой степенью достоверности

Рисунок - Главная форма

При нажатии на кнопки «Читатели», «Книги» или «Выдачи» открывается форма для редактирования, добавления или удаления записей.

В качестве примера возьмем форму «Выдачи». При нажатии на кнопку «Удалить», удалится строка с выбранной ячейкой, и выборе любой ячейки и при нажатии на кнопку «Изменить» данные в выбранной ячейке заменяются на данные из текстовых полей, при нажатии на кнопку «Добавить» добавляется новая запись с данными из текстовых полей.Изображение выглядит как снимок экрана, монитор, стена, компьютер

Описание создано с очень высокой степенью достоверности

Рисунок -Форма Выдачи

При нажатии в главной форме на кнопку «Итоги» откроется окно с таблицей, в которой отображаются все выдачи + дополнительные данные для книг и читателей.

Изображение выглядит как снимок экрана

Описание создано с очень высокой степенью достоверности

Рисунок -Форма Итоги

При нажатии на кнопку «Форма с подчиненной» открывается окно с двумя элементами: список и таблица. При выборе читателя в списке в таблице отобразятся все книги, которые он брал.

Изображение выглядит как снимок экрана

Описание создано с очень высокой степенью достоверности

Рисунок -Форма Итоги с подчиненной

При нажатии на кнопку «Запросы» откроется окно с меню, в котором можно выбрать интересующая информация. При выборе месяца-года и нажатия на кнопку «Самые популярные книги за данный период» таблица заполнится взятыми книгами за данный период, при нажатии на кнопки «Любимый читатель», «Самая популярная книга», «Самая не популярная книга», «Самый читаемый месяц» таблица заполняется соответственными данными.

Изображение выглядит как снимок экрана, монитор

Описание создано с очень высокой степенью достоверности

Рисунок - Форма Запросы

Изображение выглядит как снимок экрана, монитор, стена, внутренний

Описание создано с очень высокой степенью достоверности

Рисунок - Пример запросов

# Заключение

В результате данной работы были разработаны приложение на языке программирования C#, которое обладает следующими возможностями:

* считывания данных из текстовых файлов, визуализация данных в компоненте;
* редактирование записей посредством взаимодействия с таблицей данных;
* поиск записей по № паспортов книг, по № паспортов читателей, по дате выдачи;
* извлечение таких данных как: самые читаемые/нечитаемые книги, самые читающий читатели и самый читаемый месяц за все время;

Также в СУБД Excel и Access были сформированы приложения, позволяющие выполнять основные манипуляции с данными, что и в приложении, написанном на C#. Данные приложения упрощают работу с данными о читателях и книгах, и автоматизируют учет выданных и возвращенных книг.

# Литература

1. Документация к языку программирования C# - <https://docs.microsoft.com/ru-ru/dotnet/csharp/tour-of-csharp/>
2. Евдокимов П.В. C# на примерах 3-е издание – СПБ.: Наука и Техника, 2018 – 320 с., ил.
3. Агуров П.В. C#. Сборник рецептов. – СПб.: БХВ-Петербург, 2007. – 432 с., ил.
4. Сайт-форум - <https://ru.stackoverflow.com>
5. Онлайн уроки и курсы по программированию - <https://alekseygulynin.ru/>
6. Статьи и уроки по Excel и SQL, связи между БД - <http://office-menu.ru/>