**Национальный исследовательский университет Высшая школа экономики**

**Факультет компьютерных наук**

**Департамент программной инженерии**

***Контрольное домашнее задание***

***по дисциплине***

***«Программирование»***

Тема работы: Информационно-справочная система банкоматов Сбербанка Москвы.

Выполнил: студент группы БПИ151

Мартиросян Т.А.

«25» февраля 2016 г.

тел. 8 (929) 616-97-00

e-mail адрес: tomartirosyan@edu.hse.ru

Преподаватель: Подбельский В. В.

« »\_\_\_\_\_\_\_\_\_\_\_\_\_\_2016 г.

Москва, 2016 год. Модуль № «3»

Оглавление

[Условие задачи 2](#__RefHeading___Toc221_1182690256)

[Функции разрабатываемого приложения 3](#__RefHeading___Toc223_1182690256)

[Варианты использования 3](#__RefHeading___Toc225_1182690256)

[Описание интерфейса пользователя 3](#__RefHeading___Toc227_1182690256)

[Структура приложения 9](#__RefHeading___Toc229_1182690256)

[Диаграмма классов 9](#__RefHeading___Toc231_1182690256)

[Описание полей и методов 10](#__RefHeading___Toc233_1182690256)

[Распределение исходного кода по файлам проекта 11](#__RefHeading___Toc235_1182690256)

[Контрольный пример и описание результатов 11](#__RefHeading___Toc237_1182690256)

[Приложение 11](#__RefHeading___Toc239_1182690256)

# Условие задачи

Вариант 11

Программа Контрольного домашнего задания представляет собой WindowsForms

приложение и предназначена для просмотра, обработки и сохранения результатов

обработки данных из файла.

sberbank-atm.csv

1. Требования к основным классам приложения

1.1. Основная информация о Банкоматах сбербанка хранится в объектах класса

Банкомат. Набор полей класса задаётся полями файла sberbank-atm.csv, кроме

полей, содержащих информацию об адресе, представленном полем типа Адрес.

Класс Адрес находится в отношении композиции с классом Банкомат.

1.2. Класс Адрес представляет адрес, заданный полями CSV-файла: region, city,

address, installplace. Один из методов класса возвращает номер филиала Сбер-

банка, если банкомат установлен в филиале.

1.3. Дополнительные классы, необходимые для решения задачи (объявляет автор

программы).

2. Приложение должно поддерживать следующие функции

2.1. Открыть CSV-файл (\*.csv) с исходными данными и проверить корректность

данных в нём.

2.2. Загрузить данные из CSV-файла в объекты классов Банкомат, Адрес (объект

Адрес является уникальным для каждого объекта Банкомат) и др.

2.3. Отобразить данные из объектов в оконной форме.

2.4. Создать новую запись о Банкомате.

2.5. Удалить уже существующую запись о Банкомате.

2.6. Отредактировать существующую запись о Банкомате.

2.7. Отсортировать данные по полям: accepts\_money, bank\_code

2.8. Отобразить все банкоматы, установленные в филиале с указанным номером.

2.9. Отфильтровать данные по полям: american\_express, access

2.10. Сохранять результаты редактирований, сортировок и фильтраций в CSV-

файл. Режимы сохранения в файл: создание нового файла, замена содержимого

уже существующего файла, добавление сохраняемых данных к содержимому

существующего файла.

3. Требования к интерфейсу

3.1. При управлении файлом (загрузка, сохранение) использовать OpenFileDialog и

SaveFileDialog.

3.2. Для отображения данных использовать сетку DataGridView. Количество

отображаемых в сетке элементов (N) выбирается пользователем, N > 1 и не

превышает количества записей в файле sberbank-atm.csv.

4. Требования к устойчивости приложения

4.1. В случае ошибок открывания/сохранения файла или некорректных данных

программа должна выводить сообщение.

4.2. Аварийные ситуации должны обрабатываться, пользователю должны выводиться

информативные сообщения.

# Функции разрабатываемого приложения

## Варианты использования

Данное приложение может использоваться как удобную информационно-справочную систему банкоматов Москвы.

## Описание интерфейса пользователя

Приложение обладает интуитивно понятным интерфейсом.

После открытия приложения, вы можете открыть имеющийся файл .csv в котором уже есть записи о банкоматах. После чего вы можете его редактировать.

Пользователь может выбирать количество выводимых строк или же вывести все, путем нажатия на ячейку «Вывести все» и после этого открыть нужный файл.

Для добавления новой записи о банкомате, следует нажать соответствующую кнопку.

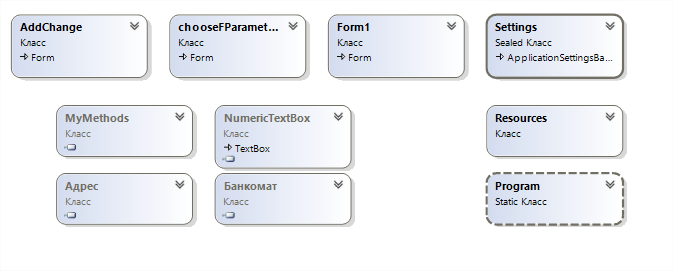
Сортировка записей осуществляется нажатием на заголовок столбца.

Для того что бы удалить или изменить запить о банкомате следует нажать правой кнопкой мыши на заголовок соответствующей строки и выбрать нужную вам опцию.

Фильтрация осуществляется выбором соответствующего пункта из вкладки меню «Вид». После фильтрации есть возможность сбросить изменения, нажав на кнопку «сбросить изменения после фильтрации»

# Структура приложения

## Диаграмма классов



## Описание полей и методов

1. Класс Form1–главный класс программы
   1. atms – поле коллекция объектов класса Банкомат
   2. backupAtms – Бекап коллекций банкоматов
   3. direction – поле задающее направление для сортировки
   4. isFilter – Флаг, покакзывающий являются ли данные отфильтрованными
   5. menuStrip – меню.
   6. accessToolStripMenuItem\_Click - Событие, при обработке которого выполняется фильтрация по полю "access"
   7. amEMenuItem4\_Click - Событие, при обработке которого выполняется фильтрация по полю "access"
   8. button1\_Click - Событие, возникающее когда пользователь нажал на кнопку, осуществляющую добавление новой записи о Банкомате
   9. dataGridView1\_ColumnHeaderMouseClick - По клику на заголовок столбца выполняется сортировка.
   10. dataGridView1\_DataBindingComplete – во время обработки этого события выбираются колонки которые можно сортировать.
   11. dataGridView1\_RowHeaderMouseClick - Обработка события при клике на заголовок строки
   12. dataGridView1\_RowPostPaint- Обработка события в котрой отрисовывается порядковый номер строки
   13. FileSave\_btn\_Click - Обработка события при желении сохранить файл путем изменения текущего файла
   14. GetAccessList - Свойство формирующее коллекцию из уникальных значений поля "access"
   15. GetAEList - Свойство формирующее коллекцию из уникальных значений поля "american express"
   16. openToolStripMenuItem\_Click- событие в котором обрабатывается открытие файла.
   17. изменитьToolStripMenuItem\_Click - Событие возникающее, если пользователь выбрал изменение строки.
   18. сохранитькакToolStripButton\_Click - Обработка события при желении сохранить файл путем создания нового файла
   19. удалитьToolStripMenuItem\_Click - Событие возникающее, если пользователь выбрал удаление строки.
   20. Atms - Возвращает коллекцию объектов класса Банкомат. Свойство только для чтения.
   21. currindex - Индекс на текущей строки с которой ведется работа
2. Класс chooseFParameter –окна для выбора параметров сортировки
   1. button1\_Click – метод применения фильтра
   2. chooseFParameter\_Load – обработка события загрузки формы
   3. FPatameter – свойство возвращающее выбранный параметр фильтра
   4. availibleParameters – поле в котором хранится коллекция доступных параметров
3. Класс AddChange- класс формы для добавления или изменения записи о банкомате
   1. AddChange - Инициализирует новый экземпляр класса AddChange предназначенный для изменения данных или добавления объектов класса Банкомат в коллекции
   2. AddChange\_Load - Обработка события загрузки формы.
   3. button1\_Click - Обработка события клика на кнопку.
4. Класс MyMethods-класс с вспомогательными методами
   1. Parserow – метод для парсинга строки \*.csv файла в коллекцию
   2. Serialization - Метод сериализации для дальнейшего сохранения
   3. SaveEncode - Метод для перевода строки в кодировку csv
   4. ShowTable – метод для изменения данных в DataTabl
   5. myTrim - Метод для обрезания боковых кавычек
5. Класс NumericTextBox- Класс наследуемый от TextBox, допускающий ввод только десятичных знаков.
   1. allowSpace - поле флаг отвечающий за недопустимолсть пробелов
   2. OnKeyPress – метод обрабатывающий, нажатие пользователем клавиши.
   3. IntValue - Возвращает значение типа Int числа введенного в NumericTextBox
   4. AllowSpace – свойство для доступа к полю allowSpace

# Распределение исходного кода по файлам проекта

Класс Form1 и все его методы находится в файле Form1.cs4

Класс chooseFParameter и все его методы находится в файле chooseFParameter.cs

Класс AddChange и все его методы находится в файле AddChange.cs

Класс MyMethods и все его методы находится в файле Class1.cs

Класс Адрес и все его методы находится в файле Адрес.cs

Класс Банкомат и все его методы находится в файле Банкомат.cs

# Контрольный пример и описание результатов

# 

Файл открыт и в таблице отображено требуемое пользователем кол-во строк.

# Текст (код) программы

ФайлForm1.cs

usingSystem;

usingSystem.Collections.Generic;

usingSystem.ComponentModel;

usingSystem.Data;

usingSystem.Drawing;

usingSystem.Linq;

usingSystem.Text;

usingSystem.Windows.Forms;

usingSystem.IO;

usingClassLibrary1;

namespaceApp

{

publicpartialclassForm1:Form

{

///<summary>

///Индекснатекущейстрокискоторойведетсяработа

///</summary>

intcurrindex{get;set;}//Автореализуемоесвойстводающеедоступкиндексустрокискотоымведетсяработа

DataTabledt=newDataTable();

///<summary>

///Имяссылкинаколлекциюобъектовкласса<seecref="Банкомат"/>

///</summary>

List<Банкомат>atms;

///<summary>

///Возвращаетколлекциюобъектовкласса<seecref="Банкомат"/>

///Свойствотолькодлячтения.

///</summary>

publicList<Банкомат>Atms

{

get

{

returnatms;

}

}

///<summary>

///Имяссылкунаколлекциюкласса<seecref="Банкомат"/>

///Предназначендлясохраненияданныхпередфильтрацией,чтобыпозволитьпользователюсброситьизменения

///</summary>

List<Банкомат>backupAtms;

///<summary>

///Флаг,показывающийявляютсялиданныеужеотфильтрованными.

///</summary>

boolisFiltered=false;

///<summary>

///Полевкоторомхранитсяпутькфайлу.

///</summary>

staticstringpath="";

///<summary>

///Направлениесортировки

///</summary>

ListSortDirectiondirection=ListSortDirection.Ascending;

publicForm1()

{

InitializeComponent();

}

///<summary>

///Событиевкотромобрабатываетсяоткрытиефайла

///</summary>

privatevoidopenToolStripMenuItem\_Click(objectsender,EventArgse)

{

if(checkBox1.Checked)numericTextBox1.Enabled=false;

elsenumericTextBox1.Enabled=true;

button1.Enabled=false;

viewToolStripMenuItem.Enabled=false;

button2.Enabled=false;

сохранитьToolStripButton.Enabled=false;

saveAsToolStripMenuItem.Enabled=false;

SaveMenuToolStripItem.Enabled=false;

сохранитькакToolStripButton.Enabled=false;

создатьToolStripButton.Enabled=false;

try

{

if(openFileDialog1.ShowDialog()==DialogResult.OK)

{

isFiltered=false;

atms=newList<Банкомат>();

dt.Rows.Clear();

dt.Columns.Clear();

dataGridView1.DataSource=dt;

path=openFileDialog1.FileName;

viewToolStripMenuItem.Enabled=true;

List<string>rows=File.ReadAllLines(path,Encoding.UTF8).ToList();

if(rows[0].Split(',').Length!=17)thrownewException();

//Заполнениезаголовковколонокзначениями.

foreach(variteminrows[0].Split(','))

{

dt.Columns.Add(item);

}

//Отрисовкатаблицывслучае,еслипользовательрешилвывестивсеэлементы.

if(checkBox1.Checked)

{

for(inti=1;i<=rows.Count-1;i++)

{

atms.Add(newБанкомат(rows[i]));

}

dataGridView1.SuspendLayout();

Cursor=Cursors.AppStarting;

MyMethods.ShowTable(atms.Count,atms,refdt);

Cursor=Cursors.Default;

//dataGridView1.DataSource=dt;

dataGridView1.ResumeLayout();

}

//Вслучееслипользовтельсамвыбралчисловвыводимыхстрок,осуществляетсяпроверкавведенныхзначений.

elseif(numericTextBox1.IntValue>0&&!string.IsNullOrWhiteSpace(numericTextBox1.Text))

{

if(numericTextBox1.IntValue>rows.Count-1)//Еслипользовательввелзначениепревышающеедопустимое.

{

if(MessageBox.Show("Запрашивоемоечислострокпревышаеткол-восуществующих."+

string.Format("\nМаксимальноечислодлявывода:{0}",rows.Count-1)+

"\nБудутвыведенывсестрокифайла.",

"Ошибкаввода.",MessageBoxButtons.OKCancel,MessageBoxIcon.Warning)==DialogResult.OK)

{

for(inti=1;i<=rows.Count-1;i++)

{

atms.Add(newБанкомат(rows[i]));

}

numericTextBox1.Text=(atms.Count).ToString();

dataGridView1.SuspendLayout();

Cursor=Cursors.AppStarting;

MyMethods.ShowTable(numericTextBox1.IntValue,atms,refdt);

Cursor=Cursors.Default;

dataGridView1.ResumeLayout();

}

else{return;}

}

else//Еслипользовательввелвсеверно.

{

for(inti=1;i<=numericTextBox1.IntValue;i++)

{

atms.Add(newБанкомат(rows[i]));

}

dataGridView1.SuspendLayout();

Cursor=Cursors.AppStarting;

MyMethods.ShowTable(numericTextBox1.IntValue,atms,refdt);

Cursor=Cursors.Default;

dataGridView1.ResumeLayout();

dataGridView1.ClearSelection();

}

}

else{

MessageBox.Show("Введенонекорректноекол-вострокдлявывода","Ошибкаввода.",

MessageBoxButtons.OK,MessageBoxIcon.Error);

}

button2.Enabled=true;

сохранитьToolStripButton.Enabled=true;

saveAsToolStripMenuItem.Enabled=true;

SaveMenuToolStripItem.Enabled=true;

сохранитькакToolStripButton.Enabled=true;

создатьToolStripButton.Enabled=true;

this.Text="BANKOMATES-"+path;

}

}

catch(ArgumentNullExceptionex)

{

numericTextBox1.Focus();

System.Media.SystemSounds.Beep.Play();

MessageBox.Show(ex.Message);

}

catch(Exception)

{

MessageBox.Show("Веротяно,выбраннедопустимыйфайл.","Ошибка",MessageBoxButtons.OK,MessageBoxIcon.Error);

}

}

privatevoidForm1\_Load(objectsender,EventArgse)

{

if(checkBox1.Checked)numericTextBox1.Enabled=false;

elsenumericTextBox1.Enabled=true;

button1.Enabled=false;

viewToolStripMenuItem.Enabled=false;

button2.Enabled=false;

}

privatevoidcheckBox1\_CheckedChanged(objectsender,EventArgse)

{

if(checkBox1.Checked)numericTextBox1.Enabled=false;

elsenumericTextBox1.Enabled=true;

}

///<summary>

///Обработкасобытиявкотройотрисовываетсяпорядковыйномерстроки

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoiddataGridView1\_RowPostPaint(objectsender,DataGridViewRowPostPaintEventArgse)

{

vargrid=senderasDataGridView;

varrowIdx=(e.RowIndex+1).ToString();

varcenterFormat=newStringFormat()

{

Alignment=StringAlignment.Center,

LineAlignment=StringAlignment.Center

};

varheaderBounds=newRectangle(e.RowBounds.Left,e.RowBounds.Top,grid.RowHeadersWidth,e.RowBounds.Height);

e.Graphics.DrawString(rowIdx,this.Font,SystemBrushes.ControlText,headerBounds,centerFormat);

}

privatevoiddataGridView1\_DataBindingComplete(objectsender,DataGridViewBindingCompleteEventArgse)

{

//Сделаемметодсортировкинужныхнамколонокпрограммным.

foreach(DataGridViewColumncolumnindataGridView1.Columns)

{

if(column.Name=="bank\_code"||column.Name=="accepts\_money")

column.SortMode=DataGridViewColumnSortMode.Programmatic;

elsecolumn.SortMode=DataGridViewColumnSortMode.NotSortable;

}

}

///<summary>

///Покликуназаголовокстолбцавыполняетсясортировка.

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoiddataGridView1\_ColumnHeaderMouseClick(

objectsender,DataGridViewCellMouseEventArgse)

{

DataGridViewColumnnewColumn=dataGridView1.Columns[e.ColumnIndex];

if(newColumn.Name=="bank\_code")

{

dataGridView1.Columns["accepts\_money"].HeaderCell.SortGlyphDirection=SortOrder.None;

if(direction==ListSortDirection.Ascending)

{

dataGridView1.SuspendLayout();

try{

atms.Sort((x,y)=>int.Parse(x.bank\_code).CompareTo(int.Parse(y.bank\_code)));

Cursor=Cursors.AppStarting;

MyMethods.ShowTable(atms.Count,atms,refdt);

}

catch(Exception){MessageBox.Show("Файлсодержитнедопустимыдлясортировкизначениявполе\"bank\_code\"","Ошибка",MessageBoxButtons.OK,MessageBoxIcon.Asterisk);}

Cursor=Cursors.Default;

//dataGridView1.DataSource=dt;

dataGridView1.ResumeLayout();

direction=ListSortDirection.Descending;

newColumn.HeaderCell.SortGlyphDirection=

direction==ListSortDirection.Ascending?

SortOrder.Descending:SortOrder.Ascending;

}

else

{

dataGridView1.SuspendLayout();

atms.Reverse();

Cursor=Cursors.AppStarting;

MyMethods.ShowTable(atms.Count,atms,refdt);

Cursor=Cursors.Default;

dataGridView1.ResumeLayout();

direction=ListSortDirection.Ascending;

newColumn.HeaderCell.SortGlyphDirection=

direction==ListSortDirection.Ascending?

SortOrder.Descending:SortOrder.Ascending;

}

}

elseif(newColumn.Name=="accepts\_money")

{

dataGridView1.Columns["bank\_code"].HeaderCell.SortGlyphDirection=SortOrder.None;

if(direction==ListSortDirection.Ascending)

{

dataGridView1.SuspendLayout();

atms.Sort((x,y)=>x.accepts\_money.CompareTo(y.accepts\_money));

Cursor=Cursors.AppStarting;

MyMethods.ShowTable(atms.Count,atms,refdt);

Cursor=Cursors.Default;

//dataGridView1.DataSource=dt;

dataGridView1.ResumeLayout();

direction=ListSortDirection.Descending;

newColumn.HeaderCell.SortGlyphDirection=

direction==ListSortDirection.Ascending?

SortOrder.Descending:SortOrder.Ascending;

}

else

{

dataGridView1.SuspendLayout();

atms.Reverse();

Cursor=Cursors.AppStarting;

MyMethods.ShowTable(atms.Count,atms,refdt);

Cursor=Cursors.Default;

//dataGridView1.DataSource=dt;

dataGridView1.ResumeLayout();

direction=ListSortDirection.Ascending;

newColumn.HeaderCell.SortGlyphDirection=

direction==ListSortDirection.Ascending?

SortOrder.Descending:SortOrder.Ascending;

}

}

}

///<summary>

///Событие,приобработкекотороговыполняетсяфильтрацияпополю"access"

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoidaccessToolStripMenuItem\_Click(objectsender,EventArgse)

{

button2.Enabled=false;

if(!isFiltered)

{

backupAtms=newList<Банкомат>();

backupAtms.AddRange(atms);

isFiltered=true;

}

chooseFParameterchoosfilterparameter=newchooseFParameter("Отфильтроватьпоaccess",this);

choosfilterparameter.ShowDialog();

if(choosfilterparameter.DialogResult==DialogResult.OK)

{

atms.RemoveAll((x)=>{if(x.access!=choosfilterparameter.FPatameter)returntrue;elsereturnfalse;});

choosfilterparameter.Close();

choosfilterparameter.Dispose();

Cursor=Cursors.AppStarting;

MyMethods.ShowTable(atms.Count,atms,refdt);

Cursor=Cursors.Default;

button1.Enabled=true;

}

}

///<summary>

///Событие,приобработкекотороговыполняетсяфильтрацияпополю"american\_express"

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoidamEMenuItem4\_Click(objectsender,EventArgse)

{

button2.Enabled=false;

if(!isFiltered)

{

backupAtms=newList<Банкомат>();

backupAtms.AddRange(atms);

isFiltered=true;

}

chooseFParameterchoosfilterparameter=newchooseFParameter("Отфильтроватьпоamerican\_express",this);

choosfilterparameter.ShowDialog();

if(choosfilterparameter.DialogResult==DialogResult.OK)

{

atms.RemoveAll((x)=>{if(x.american\_express!=choosfilterparameter.FPatameter)returntrue;elsereturnfalse;});

choosfilterparameter.Close();

choosfilterparameter.Dispose();

Cursor=Cursors.AppStarting;

MyMethods.ShowTable(atms.Count,atms,refdt);

Cursor=Cursors.Default;

button1.Enabled=true;

}

}

///<summary>

///Свойствоформирующееколлекциюизуникальныхзначенийполя"access"

///</summary>

///<returns>Возвращаетколлекциютипа<seecref="List{String}"/></returns>

publicList<string>GetAccessList()

{

List<string>acc=newList<string>();

for(inti=0;i<atms.Count;i++)

{

if(!acc.Contains(atms[i].access))

{

acc.Add(atms[i].access);

}

}

returnacc;

}

///<summary>

///Свойствоформирующееколлекциюизуникальныхзначенийполя"americanexpress"

///</summary>

///<returns>Возвращаетколлекциютипа<seecref="List{String}"/></returns>

publicList<string>GetAEList()

{

List<string>ae=newList<string>();

for(inti=0;i<atms.Count;i++)

{

if(!ae.Contains(atms[i].american\_express))

{

ae.Add(atms[i].american\_express);

}

}

returnae;

}

///<summary>

///Обработкасобытиякликамышкойнакнопкудлясбросаизмененийпослефильтрации.

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoidbutton1\_Click(objectsender,EventArgse)

{

atms.Clear();

atms.AddRange(backupAtms);

this.SuspendLayout();

Cursor=Cursors.AppStarting;

MyMethods.ShowTable(atms.Count,atms,refdt);

Cursor=Cursors.Default;

this.ResumeLayout();

button1.Enabled=false;

button2.Enabled=true;

}

///<summary>

///Обработкасобытияприкликеназаголовокстроки

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoiddataGridView1\_RowHeaderMouseClick(objectsender,DataGridViewCellMouseEventArgse)

{

//Прикликеправойкнопкойоткрываетсявыпадающееменю,позволяющеепользователюсделатьдальнейшийвыбор.

if(e.Button==MouseButtons.Right)

{

contextMenuStrip1.Show(Cursor.Position);

currindex=e.RowIndex;

}

}

///<summary>

///Событиевозникающее,еслипользовательвыбралудалениестроки.

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoidудалитьToolStripMenuItem\_Click(objectsender,EventArgse)

{

intfirstindex=dataGridView1.FirstDisplayedScrollingRowIndex;

atms.RemoveAt(currindex);

Cursor=Cursors.AppStarting;

MyMethods.ShowTable(atms.Count,atms,refdt);

if(dataGridView1.RowCount>0)dataGridView1.FirstDisplayedScrollingRowIndex=firstindex;

Cursor=Cursors.Default;

}

///<summary>

///Событиевозникающее,еслипользовательвыбрализменениестроки.

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoidизменитьToolStripMenuItem\_Click(objectsender,EventArgse)

{

intfirstindex=dataGridView1.FirstDisplayedScrollingRowIndex;

AddChangeChange=newAddChange(refatms,currindex);

Change.ShowDialog();

Cursor=Cursors.AppStarting;

if(Change.DialogResult==DialogResult.OK){

Cursor=Cursors.AppStarting;

dataGridView1.SuspendLayout();

MyMethods.ShowTable(atms.Count,atms,refdt);

dataGridView1.ResumeLayout();

dataGridView1.FirstDisplayedScrollingRowIndex=firstindex;

Cursor=Cursors.Default;}

dataGridView1.FirstDisplayedScrollingRowIndex=firstindex;}

///<summary>

///Событие,возникающеекогдапользовательнажалнакнопку,осуществляющуюдобавлениеновойзаписиоБанкомате

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoidbutton2\_Click(objectsender,EventArgse){

AddChangeAddnote=newAddChange(refatms);

Addnote.ShowDialog();

if(Addnote.DialogResult==DialogResult.OK){

Cursor=Cursors.AppStarting;

dataGridView1.SuspendLayout();

MyMethods.ShowTable(atms.Count,atms,refdt);

dataGridView1.ResumeLayout();

dataGridView1.FirstDisplayedScrollingRowIndex=dataGridView1.RowCount-1;

Cursor=Cursors.Default;}}

///<summary>

///Обработкасобытияприжелениисохранитьфайлпутемсозданияновогофайла

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoidсохранитькакToolStripButton\_Click(objectsender,EventArgse){

if(saveFileDialog1.ShowDialog()==DialogResult.OK){

//Перевестивкодировкуcsv

List<string>encoded=MyMethods.Serialization(atms).ToList();

stringhead="region,city,address,installplace,int\_cards\_support,sbercart,"+

"american\_express,for\_organizations,accepts\_money,prints\_onepass,access,"+

"comments,bank\_code,bank\_name,org\_id,org\_name,phone";

encoded.Insert(0,head);

File.WriteAllLines(saveFileDialog1.FileName,encoded.ToArray(),Encoding.UTF8);

path=saveFileDialog1.FileName;

this.Text="BANKOMATES"+path;

сохранитьToolStripButton.Enabled=true;}}

///<summary>

///Обработка события при желении сохранить файл путем изменения текущего файла

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoidFileSave\_btn\_Click(objectsender,EventArgse)

{

//Перевести в кодировку csv

List<string>encoded=MyMethods.Serialization(atms).ToList();

stringhead="region,city,address,installplace,int\_cards\_support,sbercart,"+

"american\_express,for\_organizations,accepts\_money,prints\_onepass,access,"+

"comments,bank\_code,bank\_name,org\_id,org\_name,phone";

encoded.Insert(0,head);

File.WriteAllLines(path,encoded.ToArray(),Encoding.UTF8);

}

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

{

MessageBox.Show("Требования:\n\n1) Поля файла должны быть отделены сиволом \",\"(запятая)\n" +

"2) Файл должен задаваться ровно 17-ю полями точно в таком порядке: region, city, address, installplace, int\_cards\_support, sbercart, " +

"american\_express,for\_organizations,accepts\_money,prints\_onepass,access," +

"comments,bank\_code,bank\_name,org\_id,org\_name,phone\n" +

"4) Если поле содержит разделяющий символ(,) или кавычки(\"), то оно само заключается в кавычки(\"), а кавычки в поле заменяются на двойные(\"\")\n" +

"5)Сортировку по полю \"bank\_code\" невозможно осуществить, если в этих полях представлены нечисловые значения. "+

"\n\nИнформация о функционале приложения:\n1)сортировска элементов осуществляется путем нажатия на заголовок столбца.\n2)Изменение и удаление осуществляется путем нажатия правой кнопкой мыши на соответсвующий загголовок строки."+

"\n3)Пользователь может сбросить изменения после сортировки путем нажатия на соответствующую кнопку.\n\nПриятного пользования :)",

"Помощь", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

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

{

MessageBox.Show("Приложение предназначено для просмотра,\n обработки и сохранения обработанных\n данных из файла формата .csv\nАвтор: Мартиросян Тигран Оганнесович\n НИУ ВШЭ 2016\nПреподаватель: Подбельский В.В.",

"О программе", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

Файл AddChange.cs

usingSystem;

usingSystem.Collections.Generic;

usingSystem.Windows.Forms;

usingClassLibrary1;

namespaceApp

{

///<summary>

///Представляетокнодляредактированияилидобавленияновыхэлементовтаблицы

///</summary>

publicpartialclassAddChange:Form

{

///<summary>

///Ссылканаколлекциюсостоящуюизобъектовкласса<seecref="Банкомат"/>

///</summary>

List<Банкомат>Blist;

///<summary>

///Индексобъектаскоторымведетсяработа

///</summary>

intcurrindex;

///<summary>

///Флаг,отвечающийзавыполняемыйпроцесс.

///0-дляизмененияобъекта,1-длядобавлениянового

///</summary>

intoption;

///<summary>

///Инициализируетновыйэкземпляркласса<seecref="AddChange"/>предназначенныйдляизмененияданныхобъектовкласса<seecref="Банкомат"/>вколлекции

///</summary>

///

///<paramname="Blist">Ссылканаколлекциюсостоящуюизобъектовкласса<seecref="Банкомат"/></param>

///<paramname="currindex">Индексобъектаскоторымведетсяработа</param>

publicAddChange(refList<Банкомат>Blist,intcurrindex)

{

this.option=0;

this.currindex=currindex;

this.Blist=Blist;

InitializeComponent();

}

///<summary>

///Инициализируетновыйэкземпляркласса<seecref="AddChange"/>предназначенныйдлясозданияновогообъектакласса<seecref="Банкомат"/>вколлекции

///</summary>

///<paramname="Blist">Ссылканаколлекциюсостоящуюизобъектовкласса<seecref="Банкомат"/></see></param>

publicAddChange(refList<Банкомат>Blist)

{

this.option=1;

this.Blist=Blist;

InitializeComponent();

}

///<summary>

///Обработкасобытиязагрузкиформы.

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoidAddChange\_Load(objectsender,EventArgse)

{

switch(option)

{

case0://Привыбраннойопциидляизменения

{

button1.Text="Изменить";

//заполнениетекстбоксовисходнымизначениями

textBox1.Text=Blist[currindex].adr.region;

textBox2.Text=Blist[currindex].adr.city;

textBox3.Text=Blist[currindex].adr.adress;

textBox4.Text=Blist[currindex].adr.installplace;

textBox5.Text=Blist[currindex].int\_cards\_support;

textBox6.Text=Blist[currindex].sbercart;

textBox7.Text=Blist[currindex].american\_express;

textBox8.Text=Blist[currindex].for\_organizations;

textBox9.Text=Blist[currindex].accepts\_money;

textBox10.Text=Blist[currindex].prints\_onepass;

textBox11.Text=Blist[currindex].access;

textBox12.Text=Blist[currindex].comments;

numericTextBox1.Text=Blist[currindex].bank\_code;

textBox13.Text=Blist[currindex].bank\_name;

numericTextBox2.Text=Blist[currindex].org\_id;

textBox14.Text=Blist[currindex].org\_name;

textBox15.Text=Blist[currindex].phone;

}

break;

case1://Привыбраннойопции"Добавить"

{

button1.Text="Добавить";

}

break;

}

}

///<summary>

///Обработкасобытиякликанакнопку.

///</summary>

///<paramname="sender"></param>

///<paramname="e"></param>

privatevoidbutton1\_Click(objectsender,EventArgse)

{

switch(option)

{

case0://изменить

{

Blist[currindex].adr.region=textBox1.Text;

Blist[currindex].adr.city=textBox2.Text;

Blist[currindex].adr.adress=textBox3.Text;

Blist[currindex].adr.installplace=textBox4.Text;

Blist[currindex].int\_cards\_support=textBox5.Text;

Blist[currindex].sbercart=textBox6.Text;

Blist[currindex].american\_express=textBox7.Text;

Blist[currindex].for\_organizations=textBox8.Text;

Blist[currindex].accepts\_money=textBox9.Text;

Blist[currindex].prints\_onepass=textBox10.Text;

Blist[currindex].access=textBox11.Text;

Blist[currindex].comments=textBox12.Text;

Blist[currindex].bank\_code=numericTextBox1.Text;

Blist[currindex].bank\_name=textBox13.Text;

Blist[currindex].org\_id=numericTextBox2.Text;

Blist[currindex].org\_name=textBox14.Text;

Blist[currindex].phone=textBox15.Text;

DialogResult=DialogResult.OK;

}

break;

case1://Добавить

{

Банкоматatm=newБанкомат();

atm.adr.region=textBox1.Text;

atm.adr.city=textBox2.Text;

atm.adr.adress=textBox3.Text;

atm.adr.installplace=textBox4.Text;

atm.int\_cards\_support=textBox5.Text;

atm.sbercart=textBox6.Text;

atm.american\_express=textBox7.Text;

atm.for\_organizations=textBox8.Text;

atm.accepts\_money=textBox9.Text;

atm.prints\_onepass=textBox10.Text;

atm.access=textBox11.Text;

atm.comments=textBox12.Text;

if(string.IsNullOrWhiteSpace(numericTextBox1.Text)||numericTextBox1.IntValue<0)//Проверканакорректностьвводав"bank\_code"

{

MessageBox.Show("Значениевполе\"bank\_code\"должнобытьположительнымцелымчислом.","Ошибкаввода!",MessageBoxButtons.OK,MessageBoxIcon.Error);

numericTextBox1.Focus();

break;

}

else

{

atm.bank\_code=numericTextBox1.Text;

}

atm.bank\_name=textBox13.Text;

if(string.IsNullOrWhiteSpace(numericTextBox2.Text)||numericTextBox2.IntValue<0)//Проверканакорректностьвводавполеorg\_id

{

MessageBox.Show("Значениевполе\"org\_id\"должнобытьположительнымцелымчислом.","Ошибкаввода!",MessageBoxButtons.OK,MessageBoxIcon.Error);

numericTextBox2.Focus();

break;

}

else

{

atm.org\_id=numericTextBox2.Text;

}

atm.org\_name=textBox14.Text;

atm.phone=textBox15.Text;

Blist.Add(atm);

DialogResult=DialogResult.OK;

}break;

}}}}

ФайлchooseFParameter.cs

usingSystem;

usingSystem.Collections.Generic;

usingSystem.Windows.Forms;

namespaceApp{

publicpartialclasschooseFParameter:Form{

List<string>availibleParameters=newList<string>();

Form1form1;

publicchooseFParameter(stringa,Form1\_form1){

this.Location=\_form1.Location;

form1=\_form1;

InitializeComponent();

this.Text=a;

}

publicstringFPatameter{

get{

returnthis.comboBox1.Text;}}

privatevoidchooseFParameter\_Load(objectsender,EventArgse){

if(this.Text=="Отфильтроватьпоamerican\_express"){

comboBox1.DataSource=form1.GetAEList();

comboBox1.SelectedIndex=0;}

else{

comboBox1.DataSource=form1.GetAccessList();

comboBox1.SelectedIndex=0;}}

privatevoidbutton1\_Click(objectsender,EventArgse){

DialogResult=DialogResult.OK;

this.Hide();

}}}

ФайлClass1.cs

usingSystem;

usingSystem.Collections.Generic;

usingSystem.Windows.Forms;

usingSystem.Globalization;

namespaceClassLibrary1

{

///<summary>

///Класснаследуемыйот<seecref="TextBox"/>осуществляющийпроверкунанедопустимыесимволы.

///</summary>

publicclassNumericTextBox:TextBox{

//Флаг,отвечающийзанедопустимостьпробелов.

boolallowSpace=false;

///<summary>

///Методобрабатывающий,нажатиепользователемклавиши.

///</summary>

///<paramname="e"></param>

protectedoverridevoidOnKeyPress(KeyPressEventArgse){

base.OnKeyPress(e);

NumberFormatInfonumberFormatInfo=System.Globalization.CultureInfo.CurrentCulture.NumberFormat;

stringdecimalSeparator=numberFormatInfo.NumberDecimalSeparator;

stringgroupSeparator=numberFormatInfo.NumberGroupSeparator;

stringnegativeSign=numberFormatInfo.NegativeSign;

stringkeyInput=e.KeyChar.ToString();

if(Char.IsDigit(e.KeyChar)){

//Десятичныечисла.

}

elseif(keyInput.Equals(decimalSeparator)||keyInput.Equals(groupSeparator)||

keyInput.Equals(negativeSign))

{

//Введеннаяклавишаявляетсяразделителемилизнаком"минус"

e.Handled=true;

System.Media.SystemSounds.Beep.Play();

}

elseif(e.KeyChar=='\b')

{

//Возвраткаретки.

}

elseif(this.allowSpace&&e.KeyChar==''){ }

else

{

//Введеннаяклавишаявляетсянедопустимой.

e.Handled=true;

System.Media.SystemSounds.Beep.Play();

}

}

///<summary>

///ВозвращаетдесятичноезначениечиславведенноговNumericTextBox

///</summary>

publicintIntValue{

get{

if(string.IsNullOrWhiteSpace(this.Text)||string.IsNullOrEmpty(this.Text)){

thrownewArgumentNullException("Кол-востроктребуемыхдлявывода.","Значениенеможетбытьпустым.");}

else

returnInt32.Parse(this.Text);}}

publicboolAllowSpace{

set{

this.allowSpace=value;}

get

{

returnthis.allowSpace;

}

}

}

publicclassMyMethods

{

///<summary>

///Методдляобрезаниябоковыхкавычек.

///</summary>

///<paramname="str"></param>

///<returns></returns>

privatestaticstringmyTrim(stringstr)

{

if(str.StartsWith("\"")&&str.EndsWith("\""))str=str.Substring(1,str.Length-2);

elsereturnstr;

returnstr;

}

///<summary>

///Методдляпарсастроки

///</summary>

///<paramname="csvRow">строкатипаCSV</param>

///<returnsname="lstFields">Возвращаетлистэлементовполученныхизстроки</returns>

staticpublicList<string>Parserow(stringcsvRow)

{

List<string>lstFields=newList<string>();

booliq=false;

stringtemp;

intst=0;

for(inti=0;i<csvRow.Length;i++)

{

if(csvRow[i]==','&&!iq)

{

temp=myTrim(csvRow.Substring(st,i-st));

lstFields.Add(temp.Replace("\"\"","\""));

st=i+1;

}

if(csvRow[i]=='"'&&!iq)iq=true;

elseif(csvRow[i]=='"'&&iq)iq=false;}

if(!string.IsNullOrEmpty(myTrim(csvRow.Substring(st).Replace("\"\"","\"")))){

temp=myTrim(csvRow.Substring(st));

lstFields.Add(temp.Replace("\"\"","\""));}

returnlstFields;}

publicstaticvoidShowTable(intN,List<Банкомат>atms,refSystem.Data.DataTabletable){

table.Rows.Clear();

for(inti=0;i<N;i++){

table.Rows.Add(atms[i].adr.region,atms[i].adr.city,atms[i].adr.adress,atms[i].adr.installplace,

atms[i].int\_cards\_support,atms[i].sbercart,atms[i].american\_express,atms[i].for\_organizations,

atms[i].accepts\_money,atms[i].prints\_onepass,atms[i].access,atms[i].comments,

atms[i].bank\_code,atms[i].bank\_name,atms[i].org\_id,atms[i].org\_name,atms[i].phone);}}

//Метод для перевода строки в кодировку csv

privatestaticstringSaveEncode(strings){

if(s.Contains(",")||s.Contains("\"")){

s="\""+s.Replace("\"","\"\"")+"\"";}

returns;}

//Методсериализациидлядальнейшегосохранения

publicstaticstring[]Serialization(List<Банкомат>atms){

string[]serializedArray=newstring[atms.Count];

for(inti=0;i<serializedArray.Length;i++){

serializedArray[i]=SaveEncode(atms[i].adr.region)+","+SaveEncode(atms[i].adr.city)+","+

SaveEncode(atms[i].adr.adress)+","+SaveEncode(atms[i].adr.installplace)+","+

SaveEncode(atms[i].int\_cards\_support)+","+SaveEncode(atms[i].sbercart)+","+

SaveEncode(atms[i].american\_express)+","+SaveEncode(atms[i].for\_organizations)+","+

SaveEncode(atms[i].accepts\_money)+","+SaveEncode(atms[i].prints\_onepass)+","+

SaveEncode(atms[i].access)+","+SaveEncode(atms[i].comments)+","+

SaveEncode(atms[i].bank\_code)+","+SaveEncode(atms[i].bank\_name)+","+

SaveEncode(atms[i].org\_id)+","+SaveEncode(atms[i].org\_name)+","+

SaveEncode(atms[i].phone);

}returnserializedArray;

}}}

Файл Адрес.cs

namespace ClassLibrary1

{

public class Адрес {

public Адрес(string \_region, string \_city, string \_address, string \_installplace){

region = \_region;

city = \_city;

adress = \_address;

installplace = \_installplace;}

public Адрес(){}

public string region { get; set; }

public string city { get; set; }

public string adress { get; set; }

public string installplace { get; set; } }

}

Файл Банкомат.cs

using System.Collections.Generic;

namespace ClassLibrary1{

public class Банкомат{

public Адрес adr;

public Банкомат(string \_region, string \_city, string \_address, string \_installplace,

string \_int\_cards\_support, string \_sbercart, string \_american\_express,

string \_for\_organizations, string \_accepts\_money, string \_prints\_onepass,

string \_access, string \_comments, string \_bank\_code, string \_bank\_name,

string \_org\_id, string \_org\_name, string \_phone) {

adr = new Адрес(\_region, \_city, \_address, \_installplace);

int\_cards\_support = \_int\_cards\_support; sbercart = \_sbercart; american\_express = \_american\_express; for\_organizations = \_for\_organizations;

accepts\_money = \_accepts\_money; prints\_onepass = \_prints\_onepass; access= \_access; comments= \_comments; bank\_code= \_bank\_code; bank\_name= \_bank\_name;

org\_id = \_org\_id; org\_name= \_org\_name; phone = \_phone; }

public Банкомат(string csvRow)

{

List<string> atmqualities = ClassLibrary1.MyMethods.Parserow(csvRow);

adr = new Адрес(atmqualities[0], atmqualities[1], atmqualities[2], atmqualities[3]);

int\_cards\_support = atmqualities[4]; sbercart = atmqualities[5]; american\_express = atmqualities[6];

for\_organizations = atmqualities[7];

accepts\_money = atmqualities[8]; prints\_onepass = atmqualities[9]; access = atmqualities[10];

comments = atmqualities[11]; bank\_code = atmqualities[12]; bank\_name = atmqualities[13];

org\_id = atmqualities[14]; org\_name = atmqualities[15]; phone = atmqualities[16];

}

public Банкомат() { adr = new Адрес(); }

public string int\_cards\_support { get; set; } public string sbercart { get; set; } public string american\_express { get; set; }

public string for\_organizations { get; set; } public string accepts\_money { get; set; }

public string prints\_onepass { get; set; } public string access { get; set; }

public string comments { get; set; } public string bank\_code { get; set; }

public string bank\_name { get; set; } public string org\_id { get; set; }

public string org\_name { get; set; } public string phone { get; set; }

}}

# Список литературы

1. MSDN, Руководство по программированию на C# - URL: <https://msdn.microsoft.com/ru-ru/library/67ef8sbd.aspx>
2. MSDN, Комментарии к XML-документации (Руководство по программированию на C#) – URL: <https://msdn.microsoft.com/ru-ru/library/b2s063f7.aspx>
3. Habrahabr, XML документация в C# -URL: <https://habrahabr.ru/post/41514/>