**введение**

В данном курсовом проекте предполагается реализовать программное обеспечение для обслуживания отделения почты с графическим интерфейсом, возможностью хранения файлов на диске и поиском.

Основное предназначение реализуемого программного продукта – учёт посылок и писем соответствующего отделения почты.

Язык, выбранный для реализации курсового проекта – C# – один из самых молодых языков программирования. В этом актуальность курсового проекта.

При решении данной задачи предполагается акцентировать внимание на графическом интерфейсе, удобство в использовании, загрузку и хранение данных на диске, графические интерфейсы для создания, редактирования, просмотра объектов, поиск объектов по различным признакам (не менее двух).

## 1 Анализ исходных данных

## 1.1 Характеристика требований на разработку

## Необходимо разработать программное обеспечение для обслуживания отделения почты со следующими требованиями:

## - Графический интерфейс основной программы. Предполагается использовать стандартные оконные формы, такие как Button, TextBox, ComboBox, label и т.д. для реализации данного требования. Предполагается использовать одно окно с набором управляющих элементов на нём, видимых или не видимых в зависимости от выбранного пункта меню.

## - Загрузка и хранение данных на диске. Загрузка файла базы будет осуществляться автоматически, при запуске программы. Сохранение изменений, внесённых в базу, будет происходить после нажатия пользователем соответствующей кнопки.

## - Графический интерфейс для создания, редактирования и просмотра данных.

## - Поиск по различным признакам. Поиск будет реализован по имени получателя, по имени отделения почты, по индексу, по трек-номеру, по цене, по отправителю, по заказчику, по весу и т.д.

- Фильтр по различным признакам. Фильтр будет реализован : для писем по индексу, а для посылок по весу и стоимости.

- Предполагается работа со следующими сущностями:

* Абстрактный класс pochta;
* posilka;
* pismo;
* zak\_pismo;

## ob\_pismo;

## ems\_pos;

## bander\_pos;

## paket\_pos.

## - Обязательно использование наследования и полиморфизма.

Язык программирования и среда разработки – C/C++, C#, MS Visual Studio 2005/2010. Платформа реализации – ОС Windows XP/Vista/7/2003/2008.

## 1.2 Анализ предметной области

Предметная область данной программы предполагает ведение учётности посылок, то есть добавление новой посылки в базу, посылки делятся на несколько типов(EMS, бандероль, мелкий пакет). Ведение учётности писем, то есть добавление нового письма в базу, письма так же делятся на несколько типов(заказное письмо, обычное письмо).

В задании предполагается создание базы данных для обслуживания отделения почты со следующим функционалом:

- Добавление данных:

* добавление заказного письма;
* добавление обычного письма;
* добавление EMS посылки;
* добавление мелкого пакета;
* добавление бандероли;

- Поиск по различным признакам;

- Фильтр по различным признакам;

- Просмотр информации:

* просмотр информации об обычном письме;
* просмотр информации о заказном письме;
* просмотр информации об EMS посылке;
* просмотр информации о мелком пакете;
* просмотр информации о бандероли.

- Работа с файлами:

* загрузка базы из файла (происходит автоматически);
* сохранение базы в файл.

**1.3 Специальное и общесистемное программное обеспечение**

При написании данного курсового проекта использовалось следующее стороннее программное обеспечение и инструментарий:

- Microsoft Visual Studio 2010 – написание программы;

- Microsoft Office Word 2010 – написание пояснительной записки, описания программы, руководства пользователя;

- Enterprise Architect 7.5 – проектирование и подготовка диаграмм.

**2 Программное проектирование**

**2.1 Разработка диаграммы прецедентов**

На разработку диаграммы прецедентов в значительной степени повлияло то требование, что она должна подробно отражать весь доступный функционал разрабатываемого программного обеспечения. Она должна подробно отражать работу с каждой сущностью предметной области. Так как предметная область разрабатываемого ПО – ведение учётности почтового отделения соответствующего почтового отделения, то у пользователя будет определённый набор действий, проделываемых над такими сущностями, как заказное письмо, обычное письмо, бандероль, EMS доставка, мелкий пакет.

## Действующим лицом будет пользователь, который сможет:

* добавлять новый посылку или письмо в базу;
* просматривать информацию о существующих посылках или письмах;
* удалять письмо или посылку из базы;
* искать письмо или посылку по определённым признакам;
* сохранять изменения, внесённые в базу;
* Фильтровать письма или посылки по определенным признакам.

Добавление новой посылки или письма подразумевает под собой добавление одной из пяти имеющихся сущностей. Почтальон сможет добавить неограниченное количество заказных писем, обычных писем, бандеролей, EMS посылок, мелких пакетов, перед этим заполнив поля, в которых указываются свойства каждой из этих сущностей.

Просмотр информации также подразумевает просмотр свойств любой посылки или письма, существующего в базе. Просмотр будет происходить в удобной пользователю форме.

Поиск будет происходить по любому из полей базы. При поиске пользователь вначале вводит ключ поиска, а затем просматривает его результат посредством простого нажатия на стрелку комбобокса «Результаты поиска».

Фильтр будет происходить по полям индекс и стоимость для писем , вес и стоимость для посылок. При фильтре пользователь вначале вводит ключ фильтра, а затем просматривает его результат посредством простого нажатия на стрелку комбобокса «Результаты фильтра».

Загрузка базы из файла происходит автоматически, а сохранение по клику на соответствующую кнопку.

Диаграмма прецедентов с действиями, описанными выше, представлена в приложении A.

**2.2 Разработка диаграммы классов**

В задании требуется предусмотреть работу не менее чем с восьми сущностями. В данном курсовом проекте этими сущностями будут являться:

Класс Pochta, в котором будут описаны общие для всех свойства:

* имя;
* адрес;
* отделение почты;
* стоимость посылки.

Эти свойства будут наследовать все остальные классы, так как все они будут их содержать.

ПО пишется для обслуживания почтового отделения. Уместно будет создать класс «posilka» и класс «pismo» **–** наследников «Pochta», но содержащих описания свойств, характерных только для посылок, и только для писем. В классе «posilka» такими свойствами будут «ves» и «otkuda». В «pismo» **–** «otprav» и «otkuda». На этих сущностях и будут строиться пять остальных:

* zak\_pismo;
* ob\_pismo;
* ems\_pos;
* bander\_pos;
* pak\_pos.

Класс «zak\_pismo» будет дополнительно обладать свойством «index», «ems\_pos» **–** «track».

Планируется построить следующую иерархию классов (как было сказано выше): абстрактный класс Pochta, его наследники – «posilka» и «pismo». А на них в свою очередь базируются : заказное письмо, обычное письмо, бандероль, мелкий пакет и EMS отправление, наследующие все их свойства, и добавляющие по своему дополнительному.

Также планируется создать класс Order, в котором будут описаны методы добавления каждого из типов посылок или писем в список, содержащий эти посылки или письма и методы для чтения свойств посылок или писем из файла.

Таким образом, кроме стандартного наследования, будем иметь новую связь – композицию классов, когда для создания нового класса в одно целое объединяем несколько объектов других классов.

Диаграмма классов представлена в приложении Б.

**2.3 Разработка диаграммы деятельности**

Разрабатываемое программное обеспечение будет предоставлять пользователю определённый набор действий.

Это:

* просмотр базы;
* добавление в базу новой посылки или письма;
* поиск по определённым признакам;
* фильтр по определённым признакам
* удаление из базы посылки или письма;
* сохранение внесённых изменений.

Остановимся на каждом действии подробнее, и проследим за деятельностью программы на каждом из них.

При нажатии на одну из пяти кнопок просмотра информации о посылках или письмах, выводится меню просмотра, содержащее поля, в которых и будет отображена нужная информация. Переход от одного посылки или письма к другой будет осуществляться нажатием кнопок «вперёд/назад». При нажатии на кнопку «добавить», программа отображает меню добавления с полями для ввода свойств. После того, как они будут заполнены, пользователю необходимо нажать кнопку «добавить» - новая посылка или письмо добавляется в конец списка, содержащего посылки или письма данного класса.

При нажатии на кнопку «поиск» отображается соответствующее меню. Затем будет считываться значение соответствующего ключа, проход по спискам посылок или писем и сравнение значений нужных полей с введённым значением. Если соответствие есть – название посылки или письма выводится в comboBox.

При нажатии на кнопку «фильтр» отображается соответствующее меню. Затем будет считываться значение соответствующего ключа, проход по спискам посылок или писем и сравнение значений нужных полей с введённым значением. Если соответствие есть – название посылки или письма выводится в comboBox.

Нажата кнопка «сохранить» **–** в каждый из файлов, содержащий информацию о каждом из классов посылок или писем, построчно перезаписывается текущее состояние базы.

Диаграмма деятельности представлена в приложении В.

**2.4 Разработка диаграммы состояний**

Диаграмма состояний – хорошо известное средство описания поведения системы. Она определяет возможные состояния, в которых может находиться конкретный объект, а также процесс смены состояний объекта в результате влияния некоторых событий.

Вначале системой загружаются данные из файлов базы в списки, посредством которых и ведётся основная работа. То есть программа запущена – считывается информация из файлов, формируются списки, и выводится главное меню программы. На данном этапе программа ожидает от пользователя действий. Если выбран один из пунктов просмотра какого-либо вида посылок или писем, то программа выводит соответствующее меню просмотра. Оно также предоставляет определённый набор действий. Программа вновь отображает меню по нажатию кнопки «назад». При выборе «добавления», отображается соответствующее меню, в котором после введения пользователем информации, программа считывает её из необходимых полей, создаёт новый экземпляр нужного класса (заказное письмо, обычное письмо, бандероль, мелкий пакет или EMS посылка), и заносит его в конец списка, содержащего посылки и письма нужного нам класса. После осуществления этой процедуры программа возвращает предыдущее меню просмотра посылок или писем, и вновь ждёт действий пользователя.

При выборе поиска вначале отображается его меню. Программа ожидает ввода значения для поиска (ключа) в поле (textBox). При прочтении значения, осуществляется проход по всем посылкам или письмам и поиск соответствий, если они есть, найденные посылки или письма отображаются в результатах поиска. По нажатии «назад» вновь отображается меню программы.

При выборе фильтра вначале отображается его меню. Программа ожидает ввода значения для фильтра (ключа) в два поля (textBox). При прочтении значения, осуществляется проход по всем посылкам или письмам и отфильтровывание соответствий. По нажатии «назад» вновь отображается меню программы.

После выбора действия «сохранить» программой открываются файлы, содержащие базу магазина, и в них производится построчная перезапись текущего состояния базы.

Диаграмма состояний представлена в приложении Г.

**3 Программная реализация**

**3.1 Детальная реализация функциональных частей программного обеспечения**

**3.1.1 Реализация программного обеспечения**

**3.1.1.1 Описание методов классов**

Как уже было описано выше, иерархия классов будет следующей: класс Pochta, содержащий описания четырёх основных общих свойств (name, adress, otdelen, cost). Его наследниками будут классы pismo и posilka, которые будут дополнять его свойства парой своих. Для pismo это otprav и otkuda. Для posilka – otprav и ves. Также будет описан класс Order, в котором будут реализованы методы для добавления экземпляров нужного класса в список, и чтения свойств экземпляров этого класса из файла.

В каждом классе сущностей описаны методы для осуществления доступа к свойствам класса, конструктор данного класса.

Разберем вышеописанное на примере класса pismo(см. листинг 3.1.1.1.1).

*Листинг 3.1.1.1.1 – Методы доступа к свойствам класса*

// методы доступа к свойствам класса

private string \_otrpav; //отправитель

private string \_otkuda; // откуда отправлено

//перегруженные методы доступа из абстрактного класса pochta

public override string name

{

get

{

return \_name;

}

set

{

\_name = value;

}

}

public override string adress

{

get

{

return \_adress;

}

set

{

\_adress = value;

}

}

public override string otdelen

{

get

{

return \_otdelen;

}

set

{

\_otdelen = value;

}

}

public override int cost

{

get

{

return \_cost;

}

set

{

\_cost = value;

}

}

public string otprav

{

get { return \_otrpav; }

set { \_otrpav = value; }

}

public string otkuda

{

get { return \_otkuda; }

set { \_otkuda = value; }

}

//В качестве базового класса принят класс pochta

public pismo(string \_name, string \_adress, string \_otdelen, int \_cost, string \_votprav, string \_votkuda)

: base(\_name, \_adress, \_otdelen, \_cost

{

otprav = \_votprav;

otkuda = \_votkuda;

}

}

Таким же образом описываются классы: posilka, ems\_pos, bander\_pos, paket\_pos, zak\_pismo и ob\_pismo. Разница заключается в описании свойств, и классе, наследником которого описываемый класс является.

Основные методы класса Order – добавление объекта нужного класса в конец списка, с которым и ведётся основная работа, и чтение из файла свойств посылки или писем, и создание объекта нужного класса с этими свойствами.

Таким образом создаём список объектов «zak\_pismo»:

public List<zak\_pismo> \_zak\_pismo = new List<zak\_pismo>();

public int \_Zpcount = 0;

\_Zpcount– счётчик заказных писем данного класса.

Добавляем в список таким образом: выделяем память под объект с уже считанными параметрами, добавляем его в список посредством метода Add, и увеличиваем счётчик.

*Листинг 3.1.1.1.2 – Метод добавления объекта zak\_pismo в список*

public void dobZp(string \_name, string \_adress, string \_otdelen, int \_cost, string \_otprav, string \_otkuda, int \_index)

{

zak\_pismo \_zp = new zak\_pismo(\_name, \_adress, \_otdelen, \_cost, \_otprav, \_otkuda, \_index);

\_zak\_pismo.Add(\_zp);

\_Zpcount++;

}

Чтение из файла производится построчно, «свойство за свойством». Для объекта zak\_pismo считываются имя, адрес, отделение, стоимость, отправитель, откуда отправлено, и индекс. После считывания всех этих свойств, вызывается dobZp, в котором этот объект добавляется в список.

*Листинг 3.1.1.1.3 – Чтение свойств zak\_pismo из файла*

public void ReadZp()

{

StreamReader fi = new StreamReader("zak\_pismo.txt");

while (!fi.EndOfStream)

{

string \_name = fi.ReadLine();

string \_adress = fi.ReadLine();

string \_otdelen = fi.ReadLine();

string \_cost = fi.ReadLine();

string \_otprav = fi.ReadLine();

string \_otkuda = fi.ReadLine();

string \_index = fi.ReadLine();

dobZp(\_name, \_adress, \_otdelen, int.Parse(\_cost), \_otprav, \_otkuda, int.Parse(\_index));

}

fi.Close();

}

**3.1.1.2 Описание функционала главного окна базы**

При открытии главного окна базы происходит его инициализация, показ необходимых кнопок и пунктов меню.

Имеется набор меток, combo- и text-боксов, кнопок, которые в одних пунктах должны отображаться, а во вторых – нет.

Вначале производится инициализация формы. Задаём, какие объекты будут отображаться, а какие нет при запуске программы, а затем вызываем методы класса Order для создания списков посылок или писем (см. листинг 3.1.1.2.1).

*Листинг 3.1.1.2.1 – инициализация формы*

public Font\_image()

{

InitializeComponent();

this.Text = "Почтовое отделение";

//--- текстбоксы

\_tb.Add(tb\_test);

\_tb.Add(tb\_test2);

\_tb.Add(tb\_test3);

\_tb.Add(tb\_test4);

\_tb.Add(tb\_test5);

\_tb.Add(tb\_test6);

\_tb.Add(tb\_test7);

\_tb.Add(tb\_test8);

\_tb.Add(tb\_test9);

\_tb.Add(tb\_test10);

\_tb.Add(tb\_test11);

//----- листбоксы

\_lb.Add(lb\_test);

\_lb.Add(lb\_test2);

\_lb.Add(lb\_test3);

\_lb.Add(lb\_test4);

\_lb.Add(lb\_test5);

\_lb.Add(lb\_test6);

\_lb.Add(lb\_test7);

\_lb.Add(lb\_test8);

\_lb.Add(lb\_test9);

\_lb.Add(lb\_test10);

\_lb.Add(lb\_test11);

\_lb.Add(lb\_test12);

\_lb.Add(lb\_test13);

\_lb.Add(lb\_test14);

\_cb.Add(cb\_test);

\_cb.Add(cb\_test1);

\_cb.Add(cb\_test2);

for (int i = 0; i < 8; i++)

{

\_tb[i].Visible = false;

\_lb[i].Visible = false;

}

for (int i = 0; i < 3; i++)

{

\_cb[i].Visible = false;

}

//--- find

\_tb[8].Visible = false;

\_lb[8].Visible = false;

\_lb[9].Visible = false;

//-- filter

\_lb[10].Visible = false;

\_tb[9].Visible = false;

\_lb[11].Visible = false;

\_lb[12].Visible = false;

\_lb[13].Visible = false;

\_tb[10].Visible = false;

//------------ загружаем данные

\_order.ReadPak();

\_order.ReadBan();

\_order.ReadEms();

\_order.ReadOp();

\_order.ReadZp();

//------- - кнопки

M\_pak\_button.Visible = false;

naz.Visible = false;

vper.Visible = false;

Ban\_button.Visible = false;

Ems\_button.Visible = false;

save\_button.Visible = false;

glav\_men\_button.Visible = false;

dobavel\_button.Visible = false;

dobv\_button.Visible = false;

poisk\_button.Visible = false;

pictureBox1.Visible = false;

zakpis\_button.Visible = false;

obpis\_button.Visible = false;

dostavka\_button.Visible = true;

button1.Visible = true;

del\_button.Visible = false;

filter\_button.Visible = false;

}

Теперь рассмотрим механизм отображения заказных писем.

Вначале при клике на заказное письмо нужно определить отображение нужных элементов, что и делает метод zakpis\_button\_Click.

*Листинг 3.1.1.2.2 – отображение элементов окна при нажатии на кнопку «заказное письмо*

private void zakpis\_button\_Click(object sender, EventArgs e)

{

\_Prov = 2;

pictureBox1.Visible = true;

try

{

Image l = new Bitmap("5.jpg");

pictureBox1.Image = l;

}

catch

{

Image l = new Bitmap("not\_found\_image.jpg");

pictureBox1.Image = l;

}

naz.Visible = true;

vper.Visible = true;

save\_button.Visible = true;

poisk\_button.Visible = true;

del\_button.Visible = true;

dobv\_button.Visible = true;

for (int i = 0; i < 7; i++)

{

\_tb[i].Visible = true;

\_lb[i].Visible = true;

}

\_lb[0].Text = "Ф.И.О. получателя посылки: ";

\_tb[0].Text = \_order.\_zak\_pismo[\_Num].name;

\_lb[1].Text = "Адрес получателя посылки: ";

\_tb[1].Text = \_order.\_zak\_pismo[\_Num].adress;

\_lb[2].Text = "Отделение почты: ";

\_tb[2].Text = \_order.\_zak\_pismo[\_Num].otdelen;

\_lb[3].Text = "Стоимость посылки: ";

\_tb[3].Text = \_order.\_zak\_pismo[\_Num].cost.ToString();

\_lb[4].Text = "Отправитель: ";

\_tb[4].Text = \_order.\_zak\_pismo[\_Num].otprav;

\_lb[5].Text = "Откуда отправлена: ";

\_tb[5].Text = \_order.\_zak\_pismo[\_Num].otkuda;

\_lb[6].Text = "Индекс: ";

\_tb[6].Text = \_order.\_zak\_pismo[\_Num].index.ToString();

M\_pak\_button.Visible = false;

Ban\_button.Visible = false;

Ems\_button.Visible = false;

zakpis\_button.Visible = false;

obpis\_button.Visible = false;

dobavel\_button.Visible = false;

dostavka\_button.Visible = false;

button1.Visible = false;

glav\_men\_button.Visible = true;

filter\_button.Visible = true;

for (int i = 0; i < 3; i++)

{

\_cb[i].Visible = false;

}

//--- find

\_tb[8].Visible = false;

\_lb[8].Visible = false;

\_lb[9].Visible = false;

//-- filter

\_lb[10].Visible = false;

\_tb[9].Visible = false;

\_lb[11].Visible = false;

\_lb[12].Visible = false;

\_lb[13].Visible = false;

\_tb[10].Visible = false;

}

В зависимости от того, письмо или посылка какого класса просматривается, текстовым боксам и лейблам присваиваем текущие свойства класса и названия этих свойств.

То есть при выводе этого меню не нужно отображать кнопки для просмотра остальных классов, поиска, фильтра и т.д. Нужны лейблы и текст-боксы, т.к. они отображают информацию о свойствах посылки или письма, а также элементы управления (кнопки вперёд-назад) для этой «страницы».

bool Control.Visible как раз и определяет, отображается ли элемент управления. Здесь достаточно присвоить true или false для отображения или сокрытия элементов.

\_Prov– определяет, какой из типов посылки и письма просматривается.

\_Num – его порядковый номер. Увеличивается или уменьшается при нажатии naz или vper.

При переходе к следующему элементу коллекции всё просто: в зависимости от значения \_Prov (какая посылка или письмо просматривается) и от того, есть ли ещё посылки или письма этого типа, увеличиваем порядковый номер просматриваемой посылки или письма.

При переходе к предыдущему, если посылки или письма больше нуля, уменьшаем счётчик и обновляем отображение. Это продемонстрировано в листинге 3.1.1.2.4

*Листинг 3.1.1.2.4 – методы перехода к следующему и предыдущему элементам коллекции*

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

{

if (\_Prov == 1)

{

if (\_Num > 0)

{

\_Num--;

\_lb[0].Text = "Ф.И.О. получателя письма: ";

\_tb[0].Text = \_order.\_ob\_pismo[\_Num].name;

\_lb[1].Text = "Адрес получателя письма: ";

\_tb[1].Text = \_order.\_ob\_pismo[\_Num].adress;

\_lb[2].Text = "Отделение почты: ";

\_tb[2].Text = \_order.\_ob\_pismo[\_Num].otdelen;

\_lb[3].Text = "Стоимость письма: ";

\_tb[3].Text = \_order.\_ob\_pismo[\_Num].cost.ToString();

\_lb[4].Text = "Отправитель: ";

\_tb[4].Text = \_order.\_ob\_pismo[\_Num].otprav;

\_lb[5].Text = "Откуда отправлена: ";

\_tb[5].Text = \_order.\_ob\_pismo[\_Num].otkuda;

\_lb[6].Text = "Индекс: ";

\_tb[6].Text = \_order.\_ob\_pismo[\_Num].index.ToString();

}

else

{

System.Windows.Forms.MessageBox.Show("Вы не можете листать назад, т.к. находитесь на первом элементе!");

}

}

}

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

{

if (\_Prov == 1)

{

if ((\_order.\_Opcount - 1) > \_Num)

{

\_Num++;

\_lb[0].Text = "Ф.И.О. получателя письма: ";

\_tb[0].Text = \_order.\_ob\_pismo[\_Num].name;

\_lb[1].Text = "Адрес получателя письма: ";

\_tb[1].Text = \_order.\_ob\_pismo[\_Num].adress;

\_lb[2].Text = "Отделение почты: ";

\_tb[2].Text = \_order.\_ob\_pismo[\_Num].otdelen;

\_lb[3].Text = "Стоимость письма: ";

\_tb[3].Text = \_order.\_ob\_pismo[\_Num].cost.ToString();

\_lb[4].Text = "Отправитель: ";

\_tb[4].Text = \_order.\_ob\_pismo[\_Num].otprav;

\_lb[5].Text = "Откуда отправлена: ";

\_tb[5].Text = \_order.\_ob\_pismo[\_Num].otkuda;

\_lb[6].Text = "Индекс: ";

\_tb[6].Text = \_order.\_ob\_pismo[\_Num].index.ToString();

}

else

{

System.Windows.Forms.MessageBox.Show("Вы не можете листать вперед, т.к. находитесь на последнем элементе!");

}

}

}

При нажатии кнопоки «возвращения в главное меню» как уже было сказано выше, просто устанавливаем, какие кнопки, метки или текст-боксы будут отображены, а какие – нет.

При добавлении письма или посылки в базу в начале «разберёмся» с отображением необходимых элементов, а затем вызываем button\_addItem\_Click, где в зависимости от добавляемой посылки или письма вначале увеличивается счётчик его количества, а затем вызывается метод добавления посылки или письма, описанный в Order.cs.

*Листинг 3.1.1.2.5 – методы добавления или удаления посылки или письма в базу*

private void dobv\_button\_Click(object sender, EventArgs e)

{

naz.Visible = false;

vper.Visible = false;

dobv\_button.Visible = false;

del\_button.Visible = false;

save\_button.Visible = false;

poisk\_button.Visible = false;

filter\_button.Visible = false;

dobavel\_button.Visible = true;

if (\_Prov == 1 || \_Prov == 2)

{

for (int i = 0; i < 7; i++)

{

\_tb[i].Text = "";

}

}

if (\_Prov == 4 || \_Prov == 3 || \_Prov == 5)

{

for (int i = 0; i < 8; i++)

{

\_tb[i].Text = "";

}

}

}

private void dobavel\_button\_Click(object sender, EventArgs e)

{

if (\_Prov == 1)

{

if (\_tb[0].Text == "" || \_tb[1].Text == "" || \_tb[2].Text == "" || \_tb[3].Text == "" || \_tb[4].Text == "" ||

\_tb[5].Text == "" || \_tb[6].Text == "")

{

System.Windows.Forms.MessageBox.Show("Одно из полей пусто!");

}

else

{

\_Num++;

\_order.dobOp(\_tb[0].Text, \_tb[1].Text, \_tb[2].Text, int.Parse(\_tb[3].Text), \_tb[4].Text, \_tb[5].Text, int.Parse(\_tb[6].Text));

for (int i = 0; i < 7; i++)

{

\_tb[i].Text = "";

}

System.Windows.Forms.MessageBox.Show("Элемент добавлен в список писем с типом обычное письмо!");

for (int i = 0; i < 8; i++)

{

\_tb[i].Visible = false;

\_lb[i].Visible = false;

}

glav\_men\_button.Visible = true;

obpis\_button.Visible = true;

zakpis\_button.Visible = true;

Ems\_button.Visible = true;

Ban\_button.Visible = true;

M\_pak\_button.Visible = true;

dobavel\_button.Visible = false;

dobv\_button.Visible = false;

save\_button.Visible = false;

del\_button.Visible = false;

poisk\_button.Visible = false;

naz.Visible = false;

vper.Visible = false;

pictureBox1.Visible = false;

filter\_button.Visible = false;

for (int i = 0; i < 3; i++)

{

\_cb[i].Visible = false;

}

//--- find

\_tb[8].Visible = false;

\_lb[8].Visible = false;

\_lb[9].Visible = false;

//-- filter

\_lb[10].Visible = false;

\_tb[9].Visible = false;

\_lb[11].Visible = false;

\_lb[12].Visible = false;

\_lb[13].Visible = false;

\_tb[10].Visible = false;

}

}

}

При нажатии на кнопку «сохранить» открываем файл и построчно поочерёдно записываем в него свойства всех посылок или писем, тем самым, сохраняя изменения в базе.

При поиске посылок или писем по определённым свойствам поступаем следующим образом. Вначале фиксируем изменение свойств комбо-бокса. Введённое значение преобразуем в строку (для того, чтобы не происходило исключения из-за некорректных данных строка подходит идеально). А затем смотрим по всем посылкам или письмам в зависимости от того, по какому свойству ищем, подходят ли они в данном случае. Если да – добавляем в результирующий comboBox.

При фильтре посылок или писем по определённым свойствам поступаем следующим образом. Вначале фиксируем изменение свойств комбо-бокса. Введённое значение преобразуем в строку (для того, чтобы не происходило исключения из-за некорректных данных строка подходит идеально). А затем смотрим по всем посылкам или письмам в зависимости от того, по какому свойству ищем, подходят ли они в данном случае. Если да – добавляем в результирующий comboBox.

Если необходимо численное значение – преобразуем его из строки при помощи Parse.

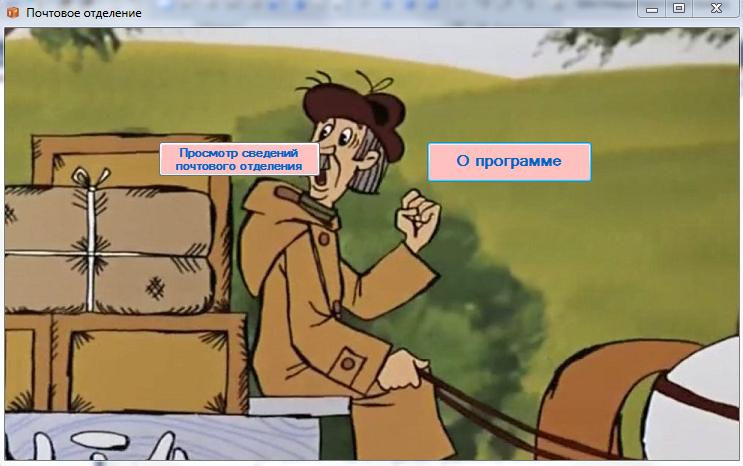
**3.1.2 Разработка пользовательского интерфейса**

Пользовательский интерфейс данной программы был разработан с использованием стандартных форм, предоставляемых средой разработки Microsoft Visual Studio 2010.

При реализации интерфейса было задействовано множество элементов. Среди них:

* ComboBox – для предоставления пользователю выбора возможных значений данных;
* Label – для текстовой отметки того или иного места, поля и т.д.;
* TextBox – для ввода произвольного текста;
* Button – кнопка;
* PictureBox – для вывода картинки.

После запуска программы открывается окно «Почтовое отделение» (см. рис. 3.1.2.1) в котором предлагается выбрать желаемое действие.

\

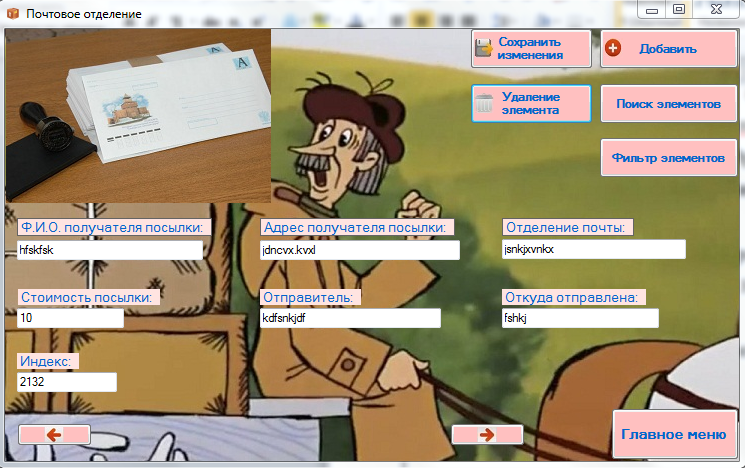
*Рис. 3.1.2.1 – Окно «Почтовое отделение»*

Изображения, показываемые при отображении свойств посылок или писем, отображаются с помощью PictureBox.

Все текстовые метки полей отображаются с помощью Label.

Для перехода на нужное действие необходимо нажать на необходимую кнопку (элемент Button).

Если пользователь выберет просмотр какого-либо или посылки, то появится следующее окно: (см. рис. 3.1.2.2).



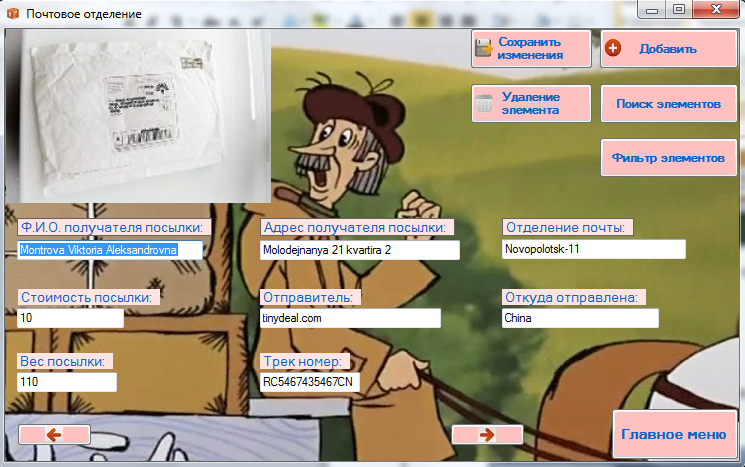
*Рис. 3.1.2.2 – Окно просмотра обычных писем*



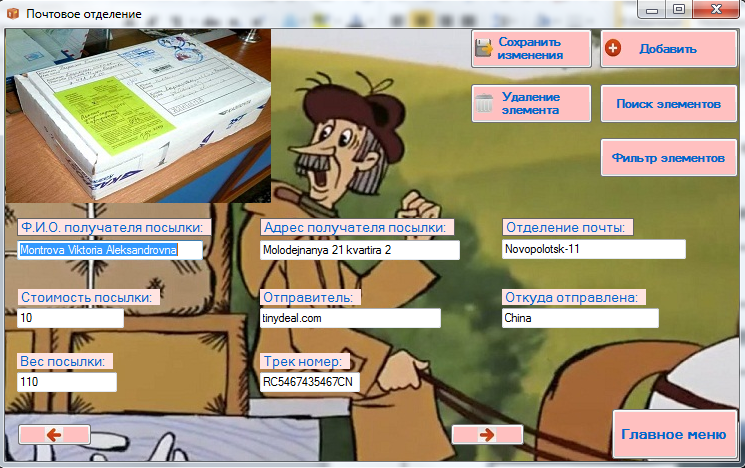
*Рис. 3.1.2.3 – Окно просмотра заказных писем*



*Рис. 3.1.2.4 – Окно просмотра EMS доставки*

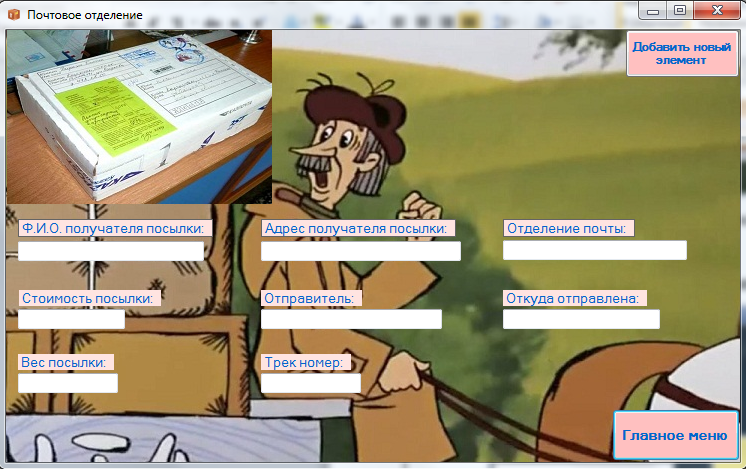


*Рис. 3.1.2.5 – Окно просмотра мелкие пакеты*



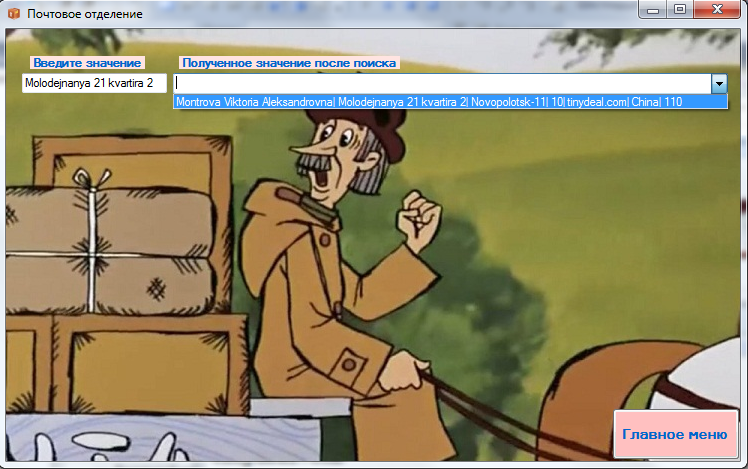
*Рис. 3.1.2.6 – Окно просмотра бандеролей*

При нажатии на кнопку «добавить», появится следующее окно:



*Рис. 3.1.2.7 – Добавление бандероли*

Окно поиска представлено на рисунке 3.1.2.8.



*Рис. 3.1.2.8 – Поиск заказного письма по определённым признакам*

**3.2 Документирование программного обеспечения**

**3.2.1 Разработка описания программы**

Данный программный продукт предназначен для обслуживания почтового отделения. Ориентирован на ведение учета заказных писем, обычных писем, бандеролей, мелких пакетов,EMS отправлений.

Для разработки использовался язык программирования C#.

Возможности ПО включают добавление данных:

* Добавление заказных писем;
* Добавление обычных писем;
* Добавление бандеролей;
* Добавление мелких пакетов;
* Добавление EMS отправлений.

Просмотр информации в виде окна с графическими элементами на нём:

* Просмотр информации о заказных письмах;
* Просмотр информации об обычных письмах;
* Просмотр информации о бандеролях;
* Просмотр информации о мелких пакетах;
* Просмотр информации об EMS отправлениях.

Поиск данных по всем признакам.

Фильтр данных по стоимости и индексу для писем и по стоимости и весу для посылок.

Сохранение в файл и загрузка базы из файла.

При написании программы использовались методы обработки информации, реализованные в системных библиотеках. Для создания пользовательского интерфейса были использованы стандартные возможности среды Microsoft Visual Studio 2010 создания окон приложения с использованием форм.

Выходными данными работы программы является файл с сохраненными данными.

Полное описание программы представлено в приложении Д.

**3.2.2 Разработка руководства пользователя**

При разработке данного руководства внимание в первую очередь акцентировалось на том, что пользователь может иметь минимальный навык обращения с компьютером. А значит, всё должно быть предельно просто и понятно.

Для запуска программы необходимо запустить исполняемый файл «kurs\_oop.exe». При выборе действия просмотра посылки или письма на экране появится окно с его свойствами.

Для добавления элемента в активную таблицу необходимо нажать кнопку добавления. Далее необходимо ввести нужные значения в соответствующие поля и нажать кнопку «Добавить».

Для поиска элементов, удовлетворяющих некоторому ключу, необходимо в соответствующее поле – ввести ключ, а затем просмотреть результаты совпадений, которые будут представлены ниже.

Для фильтра элементов, удовлетворяющих некоторому ключу, необходимо в соответствующее поле – ввести ключ, а затем просмотреть результаты совпадений, которые будут представлены ниже.

Сохранение элементов в файл происходит после нажатия пользователем данной кнопки, загрузка происходит автоматически.

Полное руководство пользователя представлено в приложении Е.

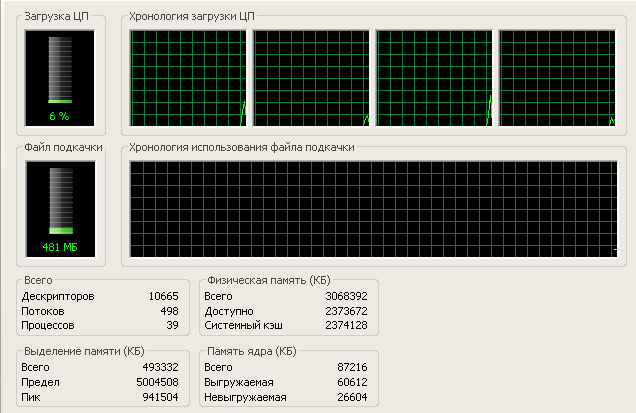
**3.2.3 Программа и методика испытаний**

При испытании данного программного продукта использовались разные средства и методы. Например, во время работы программы удалялись файлы баз данных, с которыми она работает, вводились некорректные данные при поиске,фильтре,добавлении посылки или письма. Создавались базы данных, достаточно большого объёма. Быстродействие системы (загрузка ЦП и размер файла подкачки) проверялась при помощи средств диспетчера задач.

Во время работы программы файл базы данных, содержащий информацию об одном из видов посылок был удален, однако работа приложения не нарушилась, и данные не были потеряны из-за использования структуры List.

При большой размерности базы в каждом файле возможно замедление загрузки базы при запуске программы.

На рисунке 3.2.3.1 показано быстродействие системы до запуска программы:



*Рис. 3.2.3.1 – Быстродействие системы до запуска kurs\_oop.exe*

На рисунке ниже показано быстродействие системы при работающей программе:



*Рис. 3.2.3.2 – Быстродействие системы при запущенной программе*

**3.3 Тестирование программного обеспечения**

Данные об обнаруженных ошибках и их исправлении представлены в таблице 3.3.1.

*Таблица 3.3.1 – Ошибки и отметки об их исправлении.*

|  |  |  |
| --- | --- | --- |
| Ошибка | Дата | Отметка об исправлении |
| Ошибка при поиске с пустыми параметрами | 15.12.2013 г. | Ошибка исправлена |
| Данные письма zak pismo при просмотре информации о нём отображаются некорректно, в неправильном порядке | 15.12.2013 г. | Ошибка исправлена |
| Кнопка «поиск» отображается в тех пунктах, в которых она не нужна, а при возвращении в главное меню, не отображается вовсе | 15.12.2013 г. | Ошибка исправлена |
| Кнопка «назад» отображается не во всех пунктах, которых она необходима | 16.12.2013 г. | Ошибка исправлена |
| При поиске по свойству «стоимость», он осуществляется не по всем экземплярам | 16.12.2013 г. | Ошибка исправлена |
| Необработанное исключение при поиске в случае ввода строки в поле, ожидающее ввод числа | 18.12.2013 г. | Исправлено |
| Кнопка «сохранить» не всегда отображается | 18.12.2013 г. | Ошибка исправлена |
|  | | |

**ЗАКЛЮЧЕНИЕ**

В ходе выполнения курсовой работы создано приложение «Почтовое отделение». В целом, приложение демонстрирует стабильную и правильную работу и его вполне можно использовать в реальных условиях. А именно: для учета посылок и писем в почтовом отделении.

Главным плюсом созданного приложения является то, что в нём в полном объёме реализованы задачи, которые ставились в начале разработки:

- оперативный просмотр данных;

- лёгкость использования (графический интерфейс);

- открытость системы, лёгкость в модернизации.

Были закреплены практические навыки работы с объектно-ориентированным подходом в программировании.

Были изучены основные возможности языка объектно-ориентированного программирования C#, в частности работа с окнами и использование стандартных шаблонов оконного интерфейса.

# ЛИТЕРАТУРА

1. Герберт Шилдт, Полный справочник по C#. : Пер. с англ. – М. : Издательский дом «Вильямс», 2004 – 752 с.

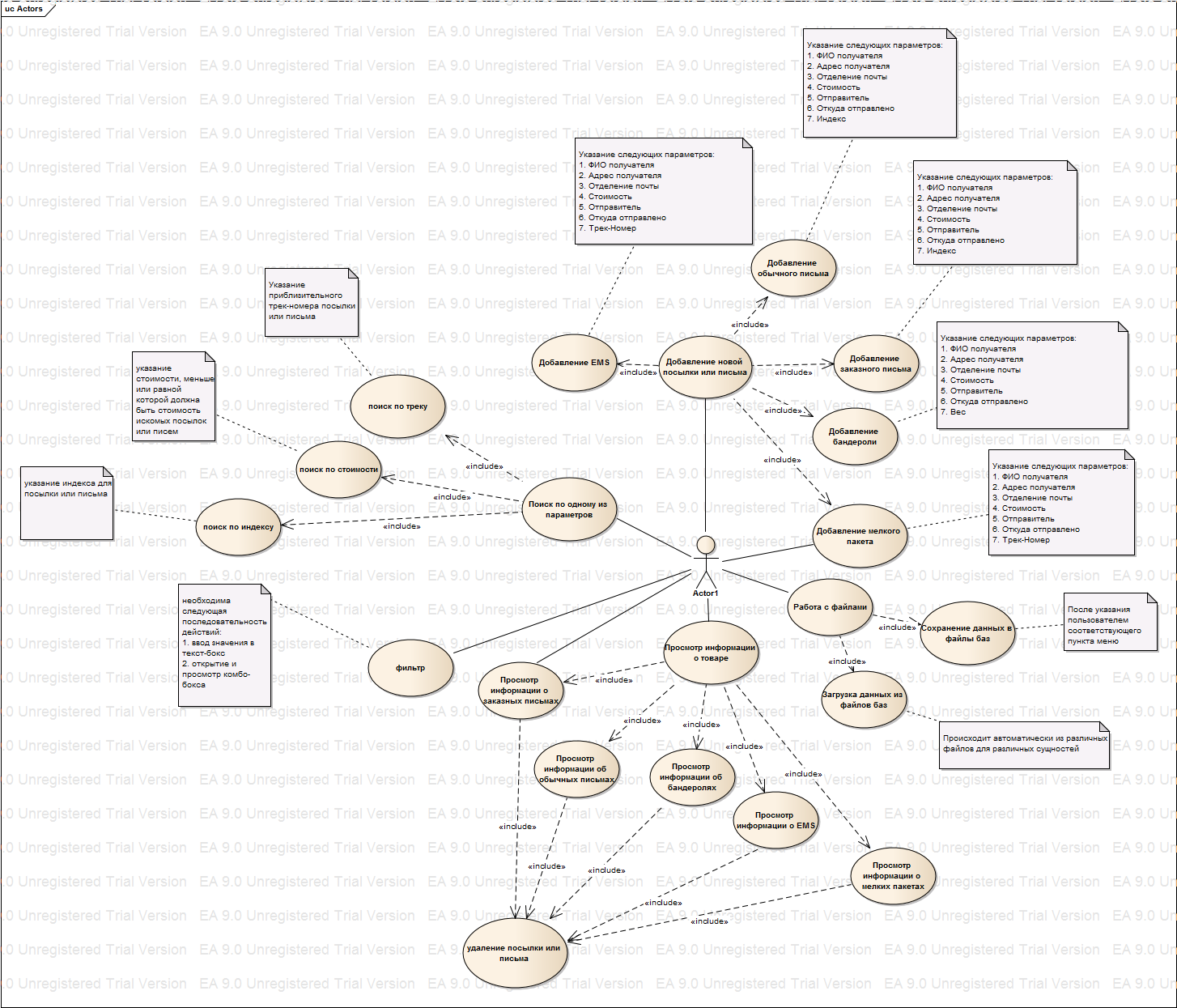
2. Дж. Бишоп, Н.Хорспул, C# в кратком изложении; Пер. с англ. – М.: БИНОМ. Лаборатория знаний, 2005 – 472 с.

3. В.В.Лабор, Си Шарп: Создание приложений для Wundows. – Мн.: Харвест, 2003 – 384 с.

4. Глухов А.О., Глухов Д.О., Методические указания к лабораторным работам по курсу «Объектно-ориентированное программирование» для студентов специальности 40.02.01, Новополцк, 2002.

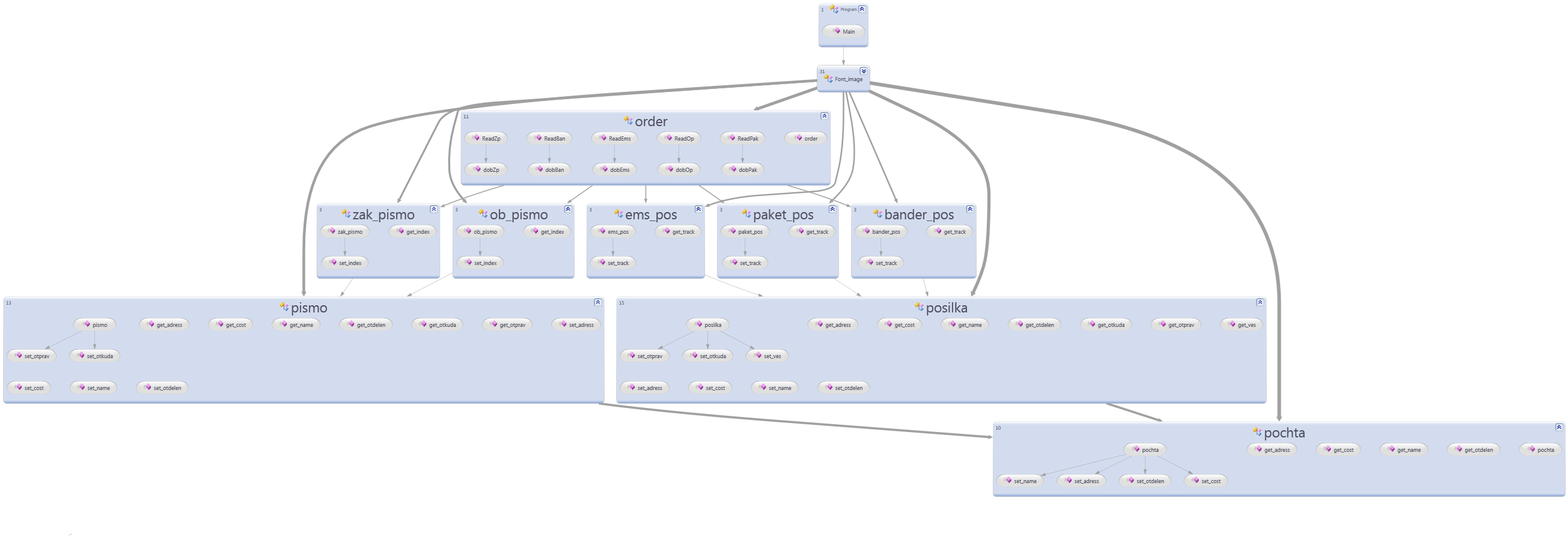
**ПРИЛОЖЕНИЕ A**

**Диаграмма прецедентов**



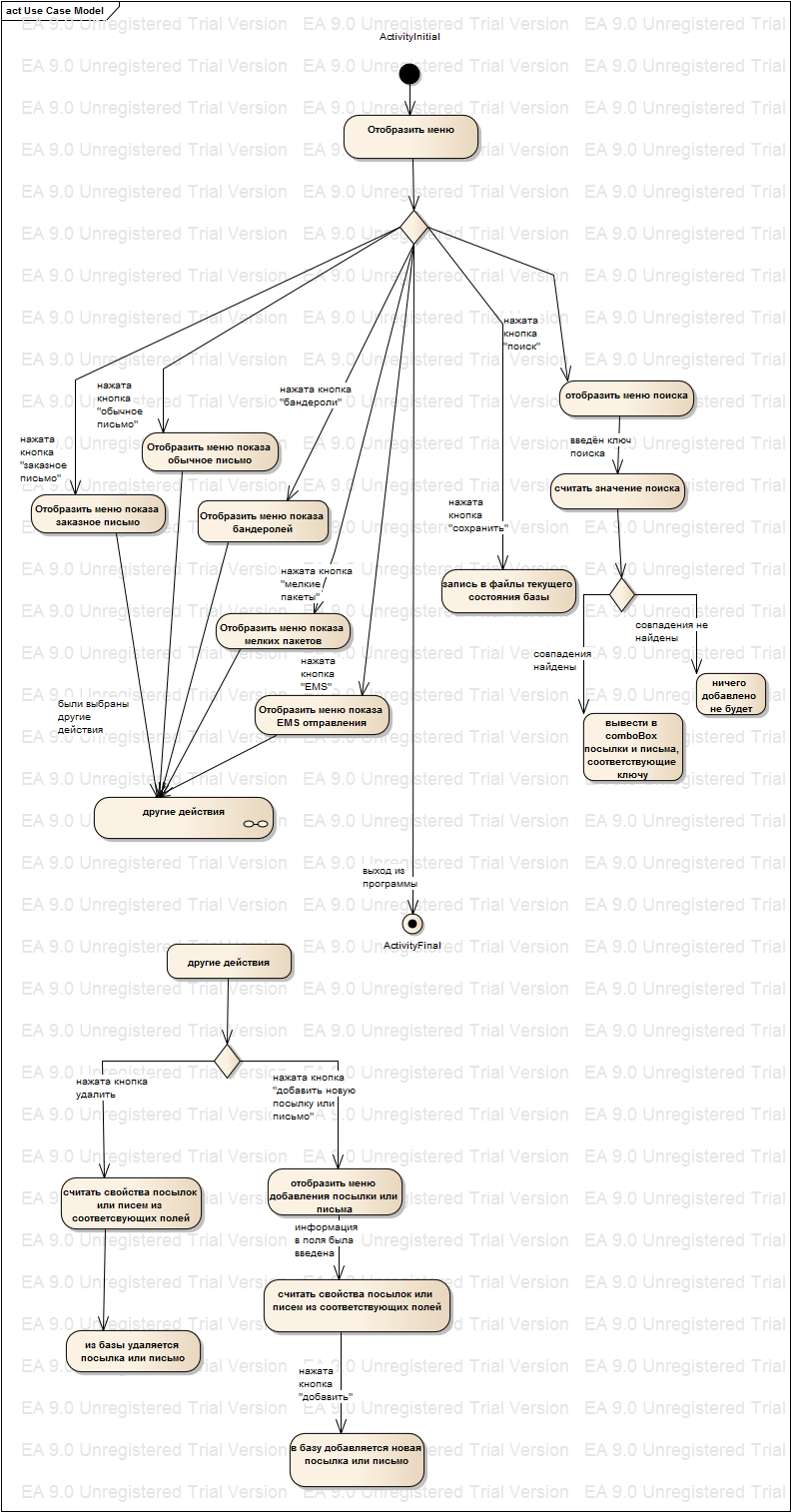
**ПРИЛОЖЕНИЕ Б**

**Диаграмма классов**



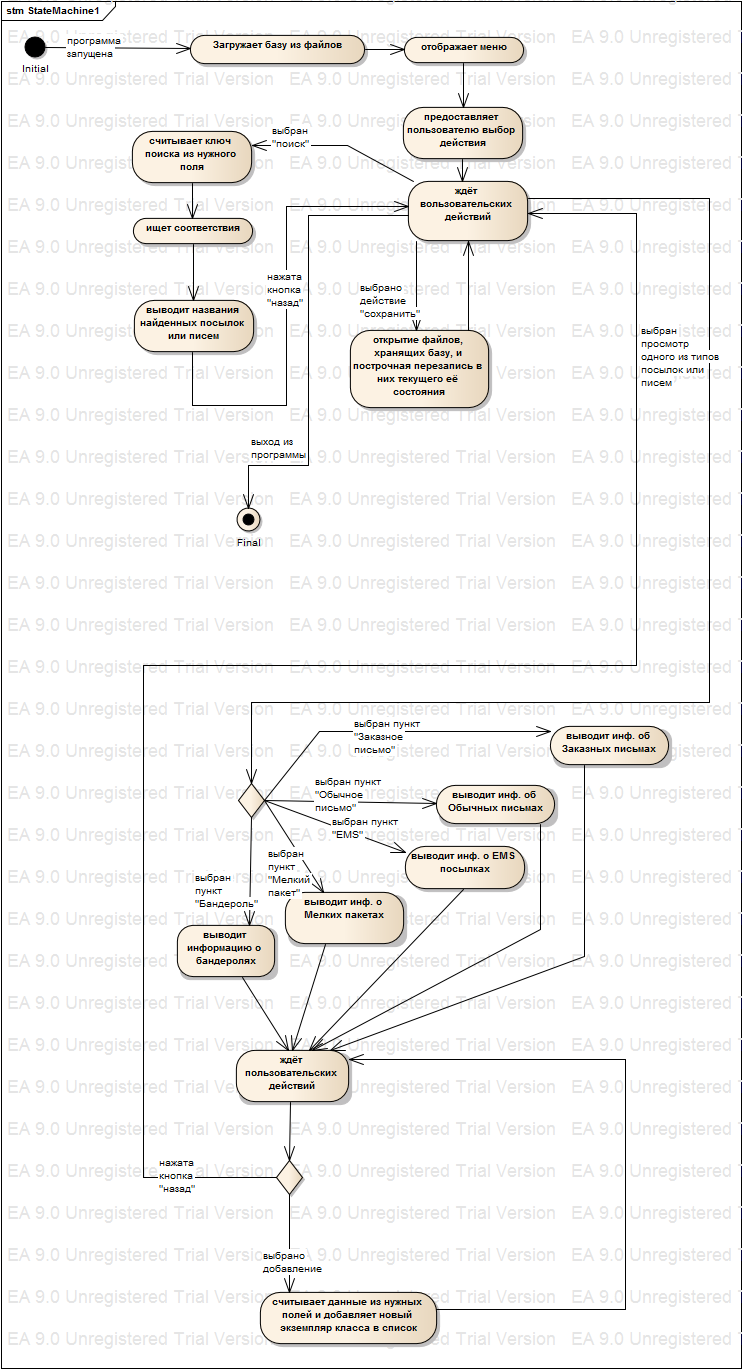
**ПРИЛОЖЕНИЕ В**

**Диаграмма взаимодействий**

****

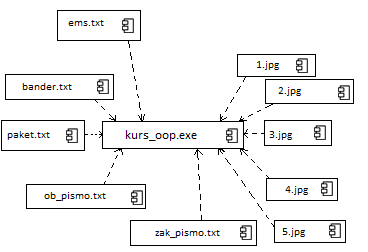
**ПРИЛОЖЕНИЕ Г**

**Диаграмма состояний**



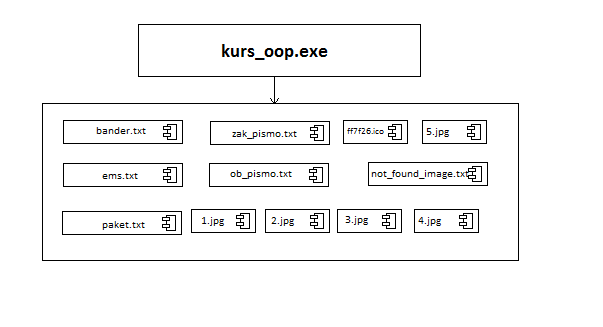
**ПРИЛОЖЕНИЕ Д**

**Диаграмма компонентов**

****

**ПРИЛОЖЕНИЕ Е**

**Диаграмма развертывания**

****

**ПРИЛОЖЕНИЕ Ж**

**Описание программы**

**Общие сведения**

Обозначение и наименование программного продукта: приложение, позволяющее вести учет заказных писем, обычных писем, бандеролей, мелких пакетов, EMS отправлений.

Наименование: «kurs\_oop».

Стороннее программное обеспечение, необходимое для функционирования программы:

- Операционная система Windows XP/Vista/7/2003/2008;

- Framework .NET 4.

Язык программирования

Для разработки данного приложения использовался язык программирования C#.

**Ж.1 Функциональное назначение**

Программа kurs\_oop предназначение для ведения учетности посылок и писем в почтовом отделении.

Ее возможности включают в себя добавление данных:

* Добавление заказных писем;
* Добавление обычных писем;
* Добавление бандеролей;
* Добавление мелких пакетов;
* Добавление EMS отправлений.

Просмотр информации в виде окна с графическими элементами на нём:

* Просмотр информации о заказных пакетах;
* Просмотр информации об обычных пакетах;
* Просмотр информации о бандеролях;
* Просмотр информации об мелких пакетах;
* Просмотр информации о EMS отправлениях.

Поиск данных по всем признакам.

Фильтр данных по стоимости и индексу для писем и по стоимости и весу для посылок.

Сохранение в файл и загрузка базы из файла.

**Ж.2 Описание логической структуры**

**Алгоритм программы**

1. При запуске программы открывается окно, в котором необходимо выбрать нужный пункт меню.

2. Нажатие одной из кнопок приводит к определённым действиям.

При выборе действия добавления элемента, появляется окно для добавления соответствующего элемента:

Считывание данных из соответствующих полей;

Создание экземпляра переменной соответствующего типа;

Заполнение значениями при помощи конструктора класса;

Добавление в структуру типа List<Type>;

При выборе действия поиска (если ключ поиска введен) происходит поиск по всем совпадениям посылок или писем.

При выборе действия фильтр (если ключ фильтра введен) происходит отфильтровывание по всем совпадениям посылок или писем.

При выборе действия просмотра посылок или писем, появляется окно с набором графических элементов, описывающих свойства данных посылок и писем.

## Используемые методы

При написании программы использовались методы обработки информации, реализованные в системных библиотеках. Для создания пользовательского интерфейса были использованы стандартные возможности среды Microsoft Visual Studio 2010 создания окон приложения с использованием форм.

## 

## Структура программы

При выборе добавления элемента в базу открывается окно для добавления соответствующего элемента. Пользователю необходимо ввести значения в предложенные поля и нажать кнопку «Добавить».

При выборе функции поиска, необходимо ввести ключ поиска.

При выборе функции фильтр, необходимо ввести ключ фильтра.

Для просмотра информации о посылке или письме, необходимо просто щёлкнуть кнопку с его названием.

Для работы данного приложения необходимо установленное ПО, указанное в пункте «Общие сведения».

**Ж.3 Используемые технические средства**

Для работы данного приложения необходим персональный компьютер (стационарный или ноутбук) со следующими минимальными системными требованиями:

* ОС Windows XP;
* 256 мб ОЗУ;
* Около 50 мб свободного места на жестком диске для установки приложения;
* Мышь;
* Монитор.

**Ж.4 Вызов и загрузка**

Программу можно запустить двойным кликом левой клавиши мыши по исполняемому файлу «kurs\_oop.exe». Если создан ярлык, существует возможность запуска через двойной клик левой клавишей мыши по ярлыку.

**Ж.5 Входные данные**

Входные данные (файлы имеют расширение .txt) должны находиться в папке с исполняемым файлов приложения.

В файлах должны храниться построчно записанные свойства каждого из типов посылок или писем.

**Ж.6 Выходные данны**е

Выходными данными работы программы является файл с сохраненными данными.

**ПРИЛОЖЕНИЕ З**

**Руководство пользователя**

**З.1 Назначение программы**

Данная программа предназначена для учета посылок и писем в почтовом отделении, а именно:

* заказных писем,
* обычных писем,
* бандеролей,
* мелких пакетах,
* EMS отправлениях.

**З.2 Условия выполнения программы**

Данная программа должна успешно функционировать на компьютерах со следующей минимальной конфигурацией:

* ОС Windows XP;
* 256 мб ОЗУ;
* 50 мб свободного места на жестком диске для установки приложения;
* Клавиатура;
* Мышь;
* Монитор.

**З.3 Выполнение программы**

**Запуск программы**

Запуск программы осуществляется при помощи исполняемого файла «kurs\_oop.exe».

При выборе действия просмотра посылки или письма на экране появится окно с его свойствами. Для добавления элемента в активную таблицу необходимо нажать кнопку добавления. Далее необходимо ввести нужные значения в соответствующие поля и нажать кнопку «Добавить».

Для поиска элементов, удовлетворяющих некоторому ключу, необходимо в соответствующее поле – ввести ключ, а затем просмотреть результаты совпадений, которые будут представлены ниже.

Для фильтра элементов, удовлетворяющих некоторому ключу, необходимо в соответствующее поле – ввести ключ, а затем просмотреть результаты совпадений, которые будут представлены ниже.

Сохранение элементов в файл происходит после нажатия пользователем данной кнопки, загрузка происходит автоматически.

**З.4 Сообщения оператору**

При работе с программой выводятся некоторые сообщения в ответ на действия оператора.