Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра электронных вычислительных машин

Дисциплина: Конструирование программ и языки программирования

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовому проекту

на тему

БАНКОВСКИЕ УСЛУГИ

БГУИР КП 1-58 01 01 25 ПЗ

Студент: гр. 110901 Русак А. Д.

Руководитель: ассистент кафедры ЭВМ   
Яночкин А. Л.

Минск 2012

Учреждение образования

«Белорусский государственный университет информатики и

радиоэлектроники»

Факультет компьютерных систем и сетей

УТВЕРЖДАЮ

Заведующий кафедрой

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2012 г.

ЗАДАНИЕ

По курсовому проектированию

Студенту *Русаку Артёму Дмитриевичу* \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

1. Тема проекта «Банковские услуги»

2. Срок сдачи студентом законченного проекта\_\_*31 декабря 2013г.\_\_*

1. 3. Исходные данные проекта. Написать программу реализации банковских услуг. Программа должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню. Работа с информацией должна производиться в окнах. В программе должна быть предусмотрена возможность создания информации в виде файлов. Программа позволяет организовать открытие нового счета, аннулирование счета, перевод денег с одного счета на другой, снятие денег со счета, депозиты и займ под проценты. Все данные содержатся в файлах. Банковский счет связывается с именем вкладчика. Необходимо реализовать оконный интерфейс при помощи ассемблерных вставок. Или реализовать ввод данных через собственные функции, реализованные на ассемблере.

4. Содержание расчетно-пояснительной записки (перечень вопросов, которые подлежат разработке): *Введение. 1.Обзор методов и алгоритмов постав-ленной задачи 2. Обоснование выбранных методов и алгоритмов 3. Описание программы для программиста 4. Описание алгоритмов решения задачи 5. Руководство пользователя. Заключение* \_

5. Перечень графического материала (с точным обозначением обязательных чертежей и графиков)

*1. Диаграмма классов*  ы

*2. Блок-схемы алгоритмов* ы

*3. Примеры работы программы* ы

6. Консультация по проекту (с обозначением разделов проекта) *Яночкин А.Л.*

7. Дата выдачи задания *14 сентября 2013 г.* –

РУКОВОДИТЕЛЬ *А.Л.* *Яночкин*

Задание принял к исполнению А.Д.Русак

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 5](#_Toc346060832)

1 Обзор МЕТОДОВ И АЛГОРИТМОВ ПОСТАВЛЕННОЙ ЗАДАЧИ 6

[1.1 Задание: 6](#_Toc346060833)

[1.2 Обзор методов и алгоритмов решения поставленной задачи: 6](#_Toc346060834)

[2 ОБОСНОВАНИЕ ВЫБРАННЫХ МЕТОДОВ 7](#_Toc346060835)

[2.1 Обоснование выбора языка программирования: 7](#_Toc346060836)

[2.2 Описание инструментов разработки программы: 7](#_Toc346060837)

[3 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА: 10](#_Toc346060838)

[3.1 Общее описание: 10](#_Toc346060839)

[3.2 Описание использованных структур данных: 10](#_Toc346060840)

[4 ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ 11](#_Toc346060841)

[5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ 13](#_Toc346060842)

[5.1 Требования к программному и аппаратному обеспечению: 13](#_Toc346060843)

[5.2Памятка клиенту: 13](#_Toc346060844)

[ЗАКЛЮЧЕНИЕ 16](#_Toc346060845)

[СПИСОК ИСПОЛЬЗАВАНЫХ ИСТОЧНИКОВ 17](#_Toc346060846)

[ПРИЛОЖЕНИЕ А- листинг программы 19](#_Toc346060847)

[ПРИЛОЖЕНИЕ Б – скриншоты работы программы 23](#_Toc346060848)

# 

# ВВЕДЕНИЕ

Сегодня на дворе 21 век - век информационных технологий, которые стремительно развиваются каждую секунду нашей жизни. Технологический, а также информационный прогресс, особенно за последние годы, сделал огромный шаг вперед. Современное человечество быстро превращается в информационное общество, а в особенности это происходит в быстроразвивающихся странах, которые делают большой упор именно на развитие техники и информационных технологий.

Почти к каждой семье имеется персональный компьютер потенциал которого огромен, использовать его возможно в различных сферах жизнедеятельности: учеба, развлечение, общение.

Также информационные технологии активно используются в сферах банковской деятельности. Пластиковые карточки, счета, виртуальные деньги. За всем этим стоит информационная система, разработанная программистами.

**1 ОБЗОР МЕТОДОВ И АЛГОРИТМОВ ПОСТАВЛЕННОЙ ЗАДАЧИ**

## 1.1 Задание

1. Написать программу реализации банковских услуг. Программа должна иметь удобный пользовательский интерфейс с необходимыми пунктами меню. Работа с информацией должна производиться в окнах. В программе должна быть предусмотрена возможность создания информации в виде файлов. Программа позволяет организовать открытие нового счета, аннулирование счета, перевод денег с одного счета на другой, снятие денег со счета, депозиты и займ под проценты. Все данные содержатся в файлах. Банковский счет связывается с именем вкладчика. Необходимо реализовать оконный интерфейс при помощи ассемблерных вставок. Или реализовать ввод данных через собственные функции, реализованные на ассемблере.

## 1.2 Обзор методов и алгоритмов решения поставленной задачи

В программе используется база данных. Средство управления базами данных (СУБД) – Microsoft SQL Server 2012. Взаимодействие происходит c помощью ORM Entity Framework. Был выбран подход Code First.

Проектирование производилось на основе архитектуры DDD(Domain-Driven Design).

# 2 ОБОСНОВАНИЕ ВЫБРАННЫХ МЕТОДОВ

## 2.1 Обоснование выбора языка программирования

Для решения поставленной задачи была выбрана интегрированная среда разработки Microsoft Visual Studio 2013.

- в ней доступны широкие возможности языка C#

- используется широким кругом программистов-профессионалов, разрабатывающих в ОС Windows и Windows Phone

- она имеет все необходимые средства и инструменты для разработки приложений любой сложности

- удобна в использовании, гибка в настройке

- создает очень эффективный код

- в ней имеется возможность использования готовых библиотек классов

- имеет встроенный механизм обмена данными по сети при помощи socket-ов

Так как ИСР выбрана Microsoft Visual Studio 2013, соответственно языком программирования выбран C#, поскольку:

- подлинная объектная ориентированность (всякая языковая сущность претендует на то, чтобы быть объектом)

- событийно-ориентированное программирование

- безопасный код (managed code)

## 2.2 Описание инструментов разработки программы

## В середине 2000 года компания Microsoft объявила о работе над новой платформой для создания программ, которая получила имя *платформа .NET* (.NET Framework). Платформа .NET образует каркас, включающий среду исполнения, библиотеку классов и набор технологий для построения приложений и служб. Основной инструмент разработки для платформы .NET – интегрированная среда Microsoft Visual Studio.

Microsoft Visual Studio 2013 – это передовое решение для разработки, позволяющее командам любого размера проектировать и создавать привлекательные приложения, которые удовлетворят самым взыскательным требованиям заказчиков. Вы можете использовать инструменты гибкого планирования — такие как планирование объема работ, панели задач и управление невыполненной работой — для внедрения методов последовательной разработки и применения гибких методологий в удобном для вас темпе. Используйте расширенные средства моделирования, обнаружения и проектирования архитектуры, чтобы описать свою систему и обеспечить полную реализацию вашей концепции архитектуры. Добейтесь взаимодействия между командами разработчиков и операторов с помощью таких программных средств, как IntelliTrace, Operations Manager Connector и Preemptive Analytics. Повышайте качество и сокращайте время устранения неполадок путем создания ошибок из развернутого программного обеспечения, включающих конкретный перечень действий, и обеспечивайте взаимодействие с операторами для получения данных, которые позволят разработчикам более детально анализировать производственные проблемы. Одним словом, вы можете создавать высококачественные инновационные решения, снижая при этом расходы на разработку.

Основой платформы .NET является *общеязыковая среда исполнения* (Common Language Runtime, CLR). CLR работает как «прослойка» между опера-ционной системой и программой для платформы .NET. Каждая программа для .NET состоит из одной или нескольких сборок. *Сборка* (assembly) является ре-зультатом компиляции исходных текстов на некотором языке программирования для платформы .NET и содержит метаданные и код на Common Intermediate Lan-guage. *Метаданные* – это информационные таблицы с полным описанием всех типов, размещённых в сборке. *Common Intermediate Language* (CIL или IL) – внутренний язык платформы .NET, он не зависит от типа процессора. В процессе работы программы CIL компилируется в машинный код специальным *JIT-ком-пилятором* (Just-in-Time compiler).

В состав платформы .NET входит обширная библиотека классов *Framework Class Library* (FCL). Частью этой библиотеки является базовый набор классов *Base Class Library* (BCL), в который входят классы для работы со строками и коллекциями данных, для поддержки многопоточности и множество других классов. В FCL также входят компоненты, поддерживающие различные технологии обработки данных и организации взаимодействия с пользователем. Это классы для работы с XML и базами данных, для создания пользовательских интерфейсов.

В стандартную поставку платформы .NET включено несколько компиляторов. Это компиляторы языков C#, F#, Visual Basic .NET, C++/CLI. Благодаря открытым спецификациям компиляторы для .NET предлагаются различными сторонними производителями. Необходимо подчеркнуть, что любой язык для платформы .NET является верхним элементом архитектуры. Имена элементов библиотеки FCL не зависят от языка программирования. Специфичной частью языка остаётся только синтаксис. Этот факт упрощает межъязыковое взаимодействие, перевод текста программы с одного языка на другой. Конечно, в синтаксисе любого языка программирования для .NET неизбежно находит своё отражение тесная связь с CLR.

**• Единая модель программирования для всех приложений** - При создании приложений ранее разработчикам приходилось использовать различные приемы программирования, которые существенным образом зависели от типа приложения — технологии разработки клиентского программного обеспечения, общедоступных веб-приложений, программного обеспечения для мобильных устройств и бизнес-логики промежуточного уровня значительно различались между собой. Интегрированная среда разработки Microsoft Visual Studio 2013 решает данную проблему, предоставляя в распоряжение разработчиков единую модель создания приложений всех категорий. Эта интегрированная модель обладает привычным и одновременно интуитивно понятным интерфейсом, позволяя разработчикам использовать свои навыки и знания для эффективного создания широкого спектра приложений.

**• Всесторонняя поддержка жизненного цикла разработки** – интегрированная среда Microsoft Visual Studio 2013 обеспечивает поддержку всего жизненного цикла разработки: начиная с планирования и проектирования через разработку и тестирование и вплоть до развертывания и последующего управления. Microsoft Visual Studio 2013 предоставляет всестороннюю адаптируемую среду для создания всех приложений, жизненно необходимых для успешной работы современных компаний.

**• Описание использованных технологий. Язык программирования C#.**

Специально для платформы .NET был разработан новый язык программирования C#. Этот язык сочетает простой синтаксис, похожий на синтаксис языков C++ и Java, и полную поддержку всех современных объектно-ориентирован-ных концепций и подходов. В качестве ориентира при разработке языка было выбрано безопасное программирование, нацеленное на создание надёжного и простого в сопровождении кода. Здесь и далее рассматривается синтаксис пятой версии языка C#, доступной в составе .NET Framework 4.5.

**C#** (произносится *си шарп*) — объектно-ориентированный язык программирования. Разработан в 1998—2001 годах группой инженеров под руководством Андерса Хейлсберга в компании Microsoft как язык разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA-334 и ISO/IEC 23270.

C# относится к семье языков с C-подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников — языков C++, Pascal, Модула, Smalltalk и в особенности Java — С#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем, например, C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

# 3 ОПИСАНИЕ ПРОГРАММЫ ДЛЯ ПРОГРАММИСТА

# 

## 3.1 Общее описание

## 

Проектирование программы производилось на основе архитектуры DDD (Domain-Driven Design).

Используется подход Entity Framework Code First. Используя этот подход, сначала описываются POCO (Playing old c-sharp classes) классы, которые являются сущностями. Сущности связаны между собой. По этим сущностям Entity Framework генерирует базу данных с соответствующими связями.

В программе 5 проектов.

1. CourseProject.Core – содержит описание доменных сущностей, интерфейсов репозиториев, интерфейса UnitOfWork.
2. CourseProject.EFData – содержит реализацию репозиториев, настройки для базы данных, реализацию UnitOfWork.
3. CourseProject.Services – содержит в себе сервисы, которые работают с репозиториями, на этом уровне производится валидация данных.
4. CourseProject.UIBank – графический интерфейс программы, на этом уровне используются сервисы.
5. CourseProject.Infrastructure – содержит инфраструктуру проекта.

Информация выводится на компонент DataGridView.

## 3.2 Описание использованных структур данных

Таблица 3.1 – Поля класса Account

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| Id | Int | Уникальный идентификатор |
| CreateDate | DateTime | Дата создания |
| Summary | Double | Сумма счёта |
| Customer | Customer | Владелец |
| CustomerId | int | Уникальный идентификатор владельца |
| Currency | Currency | Валюта |
| CurrencyId | Int | Уникальный идентификатор валюты |

Таблица 3.2 – Поля класса Currency

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| Id | int | Уникальный идентификатор |
| Value | string | Значение |

Таблица 3.3 – Поля класса Customer

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| Id | int | Уникальный идентификатор |
| Name | string | Имя |
| Surname | string | Фамилия |
| Patronymic | string | Отчество |
| BirthDate | DateTime | Дата рождения |
| Address | string | Адрес |
| Age | int | Возраст |
| PassportData | string | Паспортные данные |
| Accounts | Account | Счета владельца |
| Deposits | Deposit | Вклады владельца |
| Loans | Loan | Займы владельца |

Таблица 3.4 – Поля класса Deposit

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| Id | int | Уникальный идентификатор |
| Summary | double | Сумма вклада |
| OpeningSummary | double | Начальная сумма вклада |
| CreateDate | DateTime | Дата открытия |
| Customer | Customer | Владелец |
| CustomerId | int | Уникальный идентификатор владельца |
| Currency | Currency | Валюта |
| CurrencyId | int | Уникальный идентификатор валюты |
| OptionDeposit | OptionDeposit | Условия депозита |
| OptionDepositId | int | Уникальный идентификатор условий |

Таблица 3.5 – Поля класса Loan

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| Id | int | Уникальный идентификатор |
| Summary | double | Сумма вклада |
| OpeningSummary | double | Начальная сумма займа |
| CreateDate | DateTime | Дата открытия |
| Customer | Customer | Владелец |
| CustomerId | int | Уникальный идентификатор владельца |
| Currency | Currency | Валюта |
| CurrencyId | int | Уникальный идентификатор валюты |
| OptionLoan | OptionLoan | Условия займа |
| OptionLoanId | int | Уникальный идентификатор условий |

Таблица 3.6 – Поля класса OptionDeposit

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| Id | int | Уникальный идентификатор |
| Name | string | Имя |
| Percent | int | Проценты |
| DurationInMonth | int | Длительность в месяцах |

Таблица 3.7 – Поля класса OptionLoan

|  |  |  |
| --- | --- | --- |
| **Поле** | **Тип** | **Описание** |
| Id | int | Уникальный идентификатор |
| Name | string | Имя |
| Percent | int | Проценты |
| DurationInMonth | int | Длительность в месяцах |

# 4 ОПИСАНИЕ АЛГОРИТМОВ РЕШЕНИЯ ЗАДАЧИ

**4.1 Рассмотрим алгоритм добавления записи**

Шаг 1: Создаём временные переменные.

Шаг 2: Запускаем диалог открытия файла. Если мы получили путь, то записываем его в переменную «File\_Zap». Если нет-то завершаем работу функции.

Шаг 3: Выполняем алгоритм проверки заполнения основных полей

Шаг 4: Построчно копируем информацию из заполненных полей в соответствующие поля класса TZap

Шаг 5: Получаем переменную data используя свойство Date.FormatString компонента DateTimePicker в формате dd.mm.yyyy

Шаг 6: Записываем переменную data в соответствующее поле класса TZap.

Шаг 7: Записываем ***1***блоков по ***size*** байт каждый из области памяти, на которую указывает ***Zap*** в файл ***Fz***

Шаг 8: Очищаем используемые поля.

Шаг 9: Закрываем Доступ к файлу используя стандартную функцию fclose.

**4.2 Рассмотрим алгоритм Чтения и вывода файла на экран**

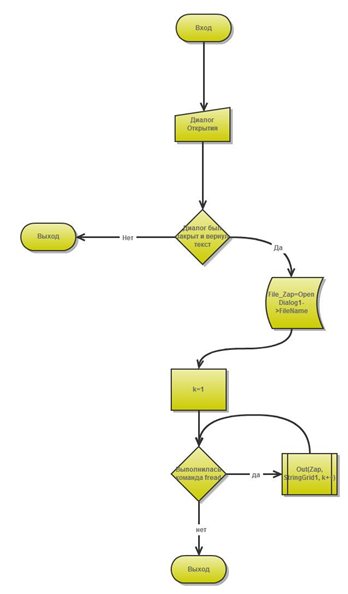


Рисунок 4.1 – Блок-схема алгоритма сохранения

# 5 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

**5.1 Требования к программному и аппаратному обеспечению**

Операционная система Windows ME/2K/XP/Seven/Ubuntu

Видео SVGA (800x600, 65536 цветов)

Процессор x86-совместимый 600 MHz

Оперативная память 128 MB

Место на жестком диске не менее 10 MB

Устройства ввода Клавиатура, Мышь

Внимание! Для корректной работы программы необходимо, чтобы в вашей системе были установлено средства .NET Framework 3.5 или более поздняя версия. Если данный компонент не установлен у Вас в системе, то его можно найти на официальном сайте Microsoft.

## 5.2 Памятка клиенту

Программа состоит из одного \*.exe файла самой программы и базы данных, которые создаются по мере её использования. В случае первого использования программы требуется создать новый файл, для этого нажав кнопку «Добавить», это вызовет окно «Добавление».

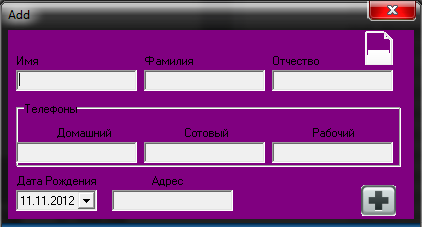


Рисунок 5.1 – Добавление Записи

В нём вводится Имя, Фамилия, Отчество, Различные телефоны такие как : домашний, рабочий, сотовый, дата рождения и адрес. После заполнения полей требуется нажать кнопку «Добавить». Это вернёт вас на главный экран. Файл будет сохранен ,пользователю останется выбрать его из списка для заполнения таблицы данными

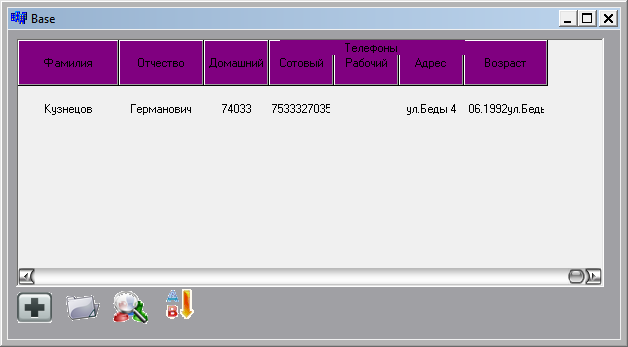


Рисунок 5.2 – Заполненная таблица

Для поиска следует нажать на кнопку «Поиск» которая перекинет вас в окно поиска

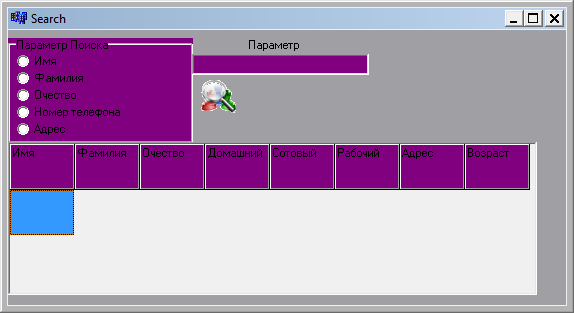


Рисунок 5.3 – Окно поиска

Для поиска требуется выбрать необходимый параметр поиска заполнить поле «Параметр» по которому будет происходить поиск и нажать на кнопку «поиск».

ЗАКЛЮЧЕНИЕ

В результате курсовой работы создана программа «Адресная Книга». Программа позволяет добавлять и сортировать данные, осушествлять поиск данных по различным параметрам. Данные загружаются вручную пользователем, сохранение производится с помощью соответствующего пункта меню.

Программный продукт выполнен в среде BorlandC++ Builder 6.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

[1] Дейтел, Х.М. Как программировать на С++ / Х.М. Дейтел, П.Д. Дейтел; пер. с англ. – М. : Бином, 2007. – 1152 с.

[2]  Страуструп, Б. Язык программирования С++ / Б. Страуструп; специальное издание. Пер. с англ. – СПб. : BHV, 2008. – 1098 с.

[3]  Скляров, В.А. Язык С++ и объектно-ориентированное программирование: справ. пособие /  В.А. Скляров. – Мн. : Вышэйшая школа, 1997. – 478 с.

[4]  Элджер, Дж. C++: библиотека программиста / Дж. Элджер. – СПб. : Питер, 2001. –  с.

[5] Шилд, Г. Программирование на Borland C++ для профессионалов / Г. Шилд.  – Мн. : ООО «Попури», 1998. – 800 с.

[6] Буч, Г. Язык UML. Руководство пользователя / Г. Буч, Д. Рамбо, А. Джекобсон; пер. с англ. – СПб. : ДМК, 2004. – 429 с.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Обозначение | | | | | | Наименование | | | | | Дополнительные  сведения | | |
|  | | | | | | Текстовые документы | | | | |  | | |
|  | | | | | |  | | | | |  | | |
| БГУИР КП 1-58 01 01 25 ПЗ | | | | | | Пояснительная записка | | | | | 17 с. | | |
|  | | | | | | Листинг программы | | | | | 12 с. | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | | Графические документы | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | | Скриншоты программы | | | | | Формат A4 | | |
|  | | | | | | Диаграмма классов | | | | | Формат A4 | | |
|  | | | | | | Блок-схемы алгоритмов | | | | | Формат A4 | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  |  | |  |  | |  |  | |  |  |  |  |  |
|  |  | |  |  | |  |  | |  |  |  |  |  |
|  |  | |  |  | |  |  | |  |  |  |  |  |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  | | | | | |  | | | | |  | | |
|  |  |  | | |  |  | | БГУИР КП 1-58 01 01 25 ПЗ | | | | | |
|  |  |  | | |  |  | |
|  |  |  | | |  |  | |
| Изм. | Л. | № Докум. | | | Подп. | Дата | | Адресная книга Ведомость курсового проекта |  | | | Лист | Листов |
| Разраб. | | Кузнецов А.Г. | | |  |  | | Т |  |  | 18 | 36 |
| Пров. | | Насуро Е.В. | | |  |  | |  | | | | |
|  | |  | | |  |  | |
|  | |  | | |  |  | |
|  | |  | | |  |  | |

# ПРИЛОЖЕНИЕ А- листинг программы

**Unit1.cpp**

**//---------------------------------------------------------------------------**

#include <vcl.h>

#pragma hdrstop

#include "Unit2.cpp"

#include "Unit1.h"

#include "Unit2.h"

#include "Unit3.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm1 \*Form1;

//---------------------------------------------------------------------------

\_\_fastcall TForm1::TForm1(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::FormCreate(TObject \*Sender)

{

StringGrid1->Cells[0][0]="Имя"; StringGrid1->Cells[1][0]="Фамилия";

StringGrid1->Cells[2][0]="Отчество"; StringGrid1->Cells[3][0]="Домашний";

StringGrid1->Cells[4][0]="Сотовый"; StringGrid1->Cells[5][0]="Рабочий";

StringGrid1->Cells[6][0]="Адрес"; StringGrid1->Cells[7][0]="Возраст";

Form1->Caption="Base";

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::StringGrid1DrawCell(TObject \*Sender, int ACol,

int ARow, TRect &Rect, TGridDrawState State)

{

TStringGrid\* StringGrid =

static\_cast<TStringGrid\*>(Sender);

StringGrid->Canvas->FillRect(Rect);

AnsiString text(

StringGrid->Cells[ACol][ARow]);

RECT RText = static\_cast<RECT>(Rect);

InflateRect(&RText, -3, -3);

DrawText(StringGrid->Canvas->Handle,

text.c\_str(), text.Length(), &RText,

DT\_CENTER | DT\_SINGLELINE |

DT\_VCENTER );

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::SpeedButton1Click(TObject \*Sender)

{

Form2->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::SpeedButton2Click(TObject \*Sender)

{

OpenDialog1->Filter=filters;

OpenDialog1->Title="Выбирете Группу";

if (OpenDialog1->Execute()){

File\_Zap=OpenDialog1->FileName;

Form3->Edit2->Text=File\_Zap;

if ((Fz=fopen(File\_Zap.c\_str(),"rb"))==NULL) {

ShowMessage("Ошибка создания ФАЙЛА!");

return;

}

int k = 1;

while(1){

if(!fread(&Zap,size,1,Fz)) break;

Out(Zap, StringGrid1, k++);

}

}

fclose(Fz);

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::SpeedButton3Click(TObject \*Sender)

{

Form3->Visible=true;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm1::SpeedButton4Click(TObject \*Sender)

{

TStringList \*SL = new TStringList;

AnsiString S;

for (int i=1; i < StringGrid1->RowCount-1; i++){

S="";

for (int j=0; j < StringGrid1->ColCount; j++) S += StringGrid1->Cells[j][i]+" ";

SL->Add(S);

}

SL->Sort();

for (int i=1; i < StringGrid1->RowCount-1; i++){

StringGrid1->Rows[i]->DelimitedText = SL->Strings[i-1];

}

delete SL;

}

//---------------------------------------------------------------------------

**Unit2.cpp**

//----------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include <io.h>

#include "Unit2.h"

#include "Unit1.h"

#include <stdio.h>

//#include "Class.h"

//----------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm2 \*Form2;

//TStringGrid \*StringGrid1;

class TZap{

public:

char name[30], lname[30], mname[30],date[10], adr[30],num1[20],num2[20],num3[20];

} Zap;

int size = sizeof(TZap);

FILE \*Fz;

AnsiString File\_Zap;

void Out(TZap, TStringGrid\*, int);

AnsiString filters = "\*.bzd";

AnsiString date;

//----------------------------------------------------------------------------

void Out(TZap z, TStringGrid \*StringGrid1, int k)

{

StringGrid1->RowCount++;

StringGrid1->Cells[0][k]=(AnsiString)Zap.name;

StringGrid1->Cells[1][k]=(AnsiString)Zap.lname;

StringGrid1->Cells[2][k]=(AnsiString)Zap.mname;

StringGrid1->Cells[3][k]=(AnsiString)Zap.num1;

StringGrid1->Cells[4][k]=(AnsiString)Zap.num2;

StringGrid1->Cells[5][k]=(AnsiString)Zap.num3;

StringGrid1->Cells[7][k]=(AnsiString)Zap.date;

StringGrid1->Cells[6][k]=(AnsiString)Zap.adr;

}

//----------------------------------------------------------------------------

\_\_fastcall TForm2::TForm2(TComponent \*Owner)

: TForm(Owner)

{

}

//----------------------------------------------------------------------------

//----------------------------------------------------------------------------

void \_\_fastcall TForm2::FormCreate(TObject \*Sender)

{

Edit1->Clear(); Edit2->Clear();Edit3->Clear();

Edit4->Clear(); Edit5->Clear();Edit6->Clear();

Edit7->Clear();

Label1->Caption="Имя"; Label2->Caption="Фамилия";

Label3->Caption="Отчество"; Label4->Caption="Домашний";

Label5->Caption="Сотовый"; Label6->Caption="Рабочий";

Label7->Caption="Дата Рождения"; Label8->Caption="Адрес";

GroupBox1->Caption="Телефоны";

Form2->Caption="Add";

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm2::SpeedButton1Click(TObject \*Sender)

{

if ((Fz = fopen(File\_Zap.c\_str(),"ab"))==NULL)

ShowMessage("Ан нет ВЫ НЕ СОЗДАЛИ ФАЙЛ!");

else{

if(Edit1->Text==""||Edit2->Text==""||Edit3->Text=="")

ShowMessage("Ан нет ВЫ ТАКИ НЕ ЗАПОЛНИЛИ ПОЛЕЙ!");

else{

strcpy( Zap.name, Edit1->Text.c\_str());

strcpy( Zap.num1, Edit4->Text.c\_str());

strcpy( Zap.num2, Edit5->Text.c\_str());

strcpy( Zap.num3, Edit6->Text.c\_str());

strcpy( Zap.lname,Edit2->Text.c\_str());

strcpy( Zap.mname,Edit3->Text.c\_str());

strcpy( Zap.adr,Edit7->Text.c\_str());

date = DateTimePicker1->Date.FormatString("dd.mm.yyyy");

strcpy( Zap.date, date.c\_str());

fwrite(&Zap, size, 1, Fz);

Edit1->Text=""; Edit2->Text=""; Edit3->Text="";

Edit4->Text=""; Edit5->Text=""; Edit6->Text=""; Edit7->Text="";

fclose(Fz); }

}

Form2->Visible = false;

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm2::SpeedButton2Click(TObject \*Sender)

{

OpenDialog1->Filter=filters;

OpenDialog1->Title="Выбирете Группу";

if (OpenDialog1->Execute()){

File\_Zap=OpenDialog1->FileName;

if ((Fz=fopen(File\_Zap.c\_str(),"ab"))==NULL) {

ShowMessage("Ошибка создания ФАЙЛА!");

return;

}

}

fclose(Fz);

}

//---------------------------------------------------------------------------

**Unit3.cpp**

//---------------------------------------------------------------------------

#include <vcl.h>

#pragma hdrstop

#include "Unit1.h"

#include "Unit1.cpp"

#include "Unit3.h"

//---------------------------------------------------------------------------

#pragma package(smart\_init)

#pragma resource "\*.dfm"

TForm3 \*Form3;

//---------------------------------------------------------------------------

\_\_fastcall TForm3::TForm3(TComponent\* Owner)

: TForm(Owner)

{

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm3::SpeedButton1Click(TObject \*Sender)

{

File\_Zap=Edit2->Text;

if ((Fz=fopen(File\_Zap.c\_str(),"rb"))==NULL)

ShowMessage("Ошибка создания ФАЙЛА!");

else {

if(Edit1->Text=="") ShowMessage("Введите искомый параметр");

else {

char q[30];

strcpy( q, Edit1->Text.c\_str());

TZap st, \*mas\_Z;

Fz = fopen(File\_Zap.c\_str(),"rb");

int D\_f = fileno(Fz); // Находим дескриптор файла

int len = filelength(D\_f); // Находим размер файла

int i, j, kol, l;

kol = len/size; //Количество записей в файле

mas\_Z = new TZap[kol]; // Считываем записи из файла в динамический массив

for (i=0; i < kol; i++)

fread((mas\_Z+i), size, 1, Fz);

fclose(Fz);

int k=1;

switch(RadioGroup1->ItemIndex){

case 0:

for (i=0; i < kol; i++)

if (!strcmp(q,mas\_Z[i].name))

{ Out( mas\_Z[i], Form3->StringGrid1,k++);

k++;

Edit1->Text="";

}

if (k==1)

ShowMessage("Not Found");

k=1;

delete []mas\_Z;

break;

case 1:

for (i=0; i < kol; i++)

if (!strcmp(q,mas\_Z[i].lname))

{ Out( mas\_Z[i], Form3->StringGrid1,k++);

k++;

Edit1->Text="";

}

if (k==1)

ShowMessage("Not Found");

k=1;

delete []mas\_Z;

break;

case 2:

for (i=0; i < kol; i++)

if (!strcmp(q,mas\_Z[i].mname))

{ Out( mas\_Z[i], Form3->StringGrid1,k++);

k++;

Edit1->Text="";

}

if (k==1)

ShowMessage("Not Found");

k=1;

delete []mas\_Z;

break;

case 3:

for (i=0; i < kol; i++)

if (!strcmp(q,mas\_Z[i].num1))

{ Out( mas\_Z[i], Form3->StringGrid1,k++);

k++;

Edit1->Text="";

}

if (k==1)

ShowMessage("Not Found");

k=1;

delete []mas\_Z;

break;

case 4:

int k=0;

for (i=0; i < kol; i++)

if (!strcmp(q,mas\_Z[i].adr))

{ Out( mas\_Z[i], Form3->StringGrid1,k++);

k++;

Edit1->Text="";

}

if (k==1)

ShowMessage("Not Found");

k=1;

delete []mas\_Z;

break;

}

}

}

}

//---------------------------------------------------------------------------

void \_\_fastcall TForm3::FormCreate(TObject \*Sender)

{

StringGrid1->Cells[0][0]="Имя"; StringGrid1->Cells[1][0]="Фамилия";

StringGrid1->Cells[2][0]="Очество"; StringGrid1->Cells[3][0]="Домашний";

StringGrid1->Cells[4][0]="Сотовый"; StringGrid1->Cells[5][0]="Рабочий";

StringGrid1->Cells[6][0]="Адрес"; StringGrid1->Cells[7][0]="Возраст";

Edit1->Clear();

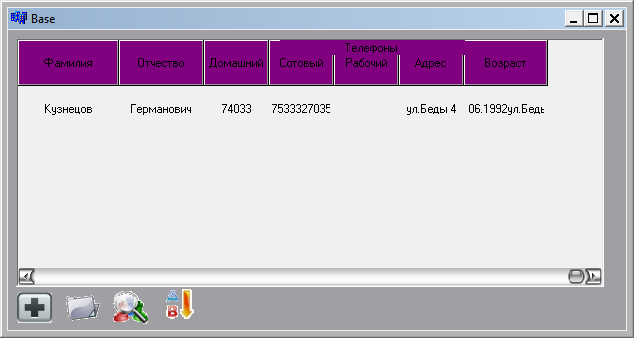
Label1->Caption="Параметр";

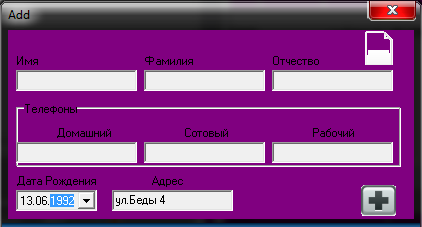
Form3->Caption="Search";

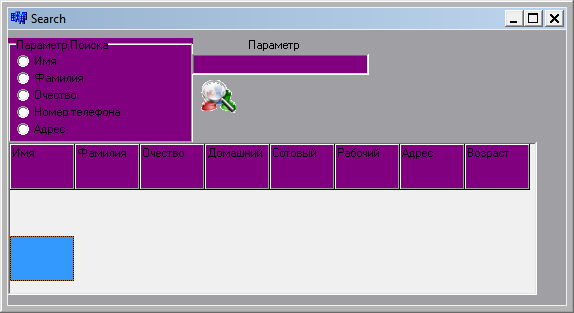
}

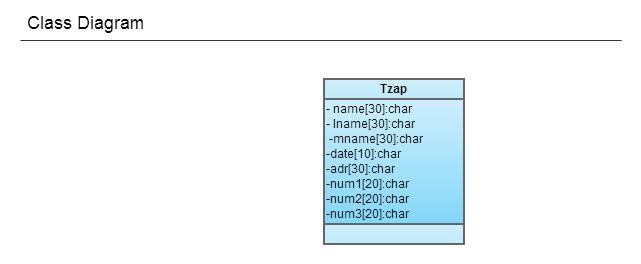
//---------------------------------------------------------------------------

# ПРИЛОЖЕНИЕ Б – скриншоты работы программы







******