Федеральное агентство по образованию Российской Федерации

Уфимский государственный авиационный технический университет

Кумертауский филиал

Кафедра ПА

КУРСОВАЯ РАБОТА

по информатике на тему: "Составление программ для решения задач

на языке программирования Turbo Pascal"

Выполнил: ст. гр. АП-109

Проверил: ст. преподаватель

Голов А.Н.

Оценка: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Подпись: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

г. Кумертау 2009

Содержание

[Введение](#_Toc250713928)

[1. Условие и постановка задачи](#_Toc250713929)

[1.1 Входная информация](#_Toc250713930)

[1.2 Выходная информация](#_Toc250713931)

[1.3 Промежуточная информация](#_Toc250713932)

[2. Формализация задачи](#_Toc250713933)

[3. Алгоритм решения задачи](#_Toc250713934)

[3.1 Словесный алгоритм](#_Toc250713935)

[3.2 Блок-схема](#_Toc250713936)

[3.3 Описание блок-схемы](#_Toc250713937)

[4. Составление программы](#_Toc250713938)

[5. Контрольный пример](#_Toc250713939)

[Заключение](#_Toc250713940)

[Список литературы](#_Toc250713941)

## Введение

Turbo Pascal - это среда разработки для языка программирования **Паскаль**. Используемый в Turbo Pascal диалект базировался на более раннем UCSD **Pascal**, получившем распространение, в первую очередь, на компьютерах серии **Apple II**. Компилирующая компонента Turbo Pascal была основана на компиляторе Blue Label Pascal, первоначально созданном в 1981 году Андерсом Хейлсбергом для операционной системы NasSys микрокомпьютера Nascom. Позднее он был переписан как Compass Pascal для операционной системы CP/M, затем как Turbo Pascal для DOS и CP/M. Одна из версий Turbo Pascal была доступна под Apple Macintosh примерно с 1986 года, но её разработка прекратилась примерно в 1992.

В 1982 году Филипп Кан приобрёл компилятор у Андерса Хейлсберга и перебрался из Парижа в Калифорнию, где основал компанию Borland.

Когда в 1983 году появилась первая версия Turbo Pascal, такой тип среды разработки был относительно новым. Во время дебюта на американском рынке, Turbo Pascal продавался по цене в 49,99 долл. Помимо привлекательной цены, встроенный компилятор Паскаля также был очень высокого качества. Приставка "Turbo" намекала как на скорость компиляции, так и на скорость производимого им исполняемого кода. Turbo Pascal создавал машинный код за один проход, без шага компоновки.

После рекламной кампании за первый месяц поступило заказов на 150 тыс. долларов - так много, что местные банки отказывались оплачивать чеки и кредитные карточки, подозревая мошенничество.

За первые два года было продано не менее 300 тысяч копий компилятора, что превзошло объём продаж всех прочих языков для микрокомпьютеров.

Для того времени это была потрясающая среда разработки. Она была проста и интуитивно понятна, с хорошо организованным меню. Ранние версии использовали раскладку горячих клавиш WordStar. В более поздних версиях появилась возможность быстро получить определение ключевого слова языка, просто поставив курсор на ключевое слово и нажав клавишу справки. Справочные статьи часто включали примеры кода, использующего данное ключевое слово. Это позволяло неопытным программистам изучать Паскаль даже без помощи книг, используя лишь среду разработки. В поставку входило большое количество исходных текстов демонстрационных и прикладных программ. В их числе были даже шахматы.

Среда позволяла легко встраивать в код на Паскале вставки на языке ассемблера. Пользователь имел возможность проходить программу шаг за шагом; при переходе на ассемблерный блок это также работало. В любой момент пользователь мог добавить переменную или регистр в удобно расположенное окно для наблюдения за ними. При построчной отладке программ, использующих графические режимы IBM PC, происходило корректное переключение между графическим режимом программы и текстовым режимом среды разработки.

Помимо всего этого, имелось средство профилирования. Книги, включённые в поставку Borland Pascal, давали детальное описание языка ассемблера Intel вплоть до указания количества тактовых циклов, необходимых для выполнения каждой инструкции. В общем и целом, система давала превосходные возможности для оптимизации кода; пользователю не требовалось пользоваться чем-либо кроме среды разработки. Всё было сделано так идеально, что даже школьник мог этим пользоваться. Эти качества позволили версии Паскаля от Borland стать стандартом языка Паскаль de facto.

С начала 1990-х TP/BP используется в университетах для изучения фундаментальных концепций программирования.

Вероятно, разработка Microsoft Pascal была прекращена из-за конкуренции с высоким качеством и небольшой ценой Turbo Pascal. Другая версия гласит, что Borland заключил соглашение с Microsoft на прекращение разработки Turbo BASIC (среды разработки для BASIC, ответвившейся от Turbo Pascal), если Microsoft прекратит разработку Microsoft Pascal. Некоторое время Microsoft выпускал QuickPascal, который был почти 100% -совместим с Turbo Pascal.

В течение нескольких лет Borland улучшал не только среду разработки, но и язык. В версии 5.5 в него были введены передовые возможности объектно-ориентированного программирования. Последней выпущенной версией была версия 7. Borland Pascal 7 включал в себя среду разработки и компиляторы для создания программ под DOS, под DOS с расширителем DOS и Windows 3. x, в то время как Turbo Pascal 7 мог создавать только обычные DOS-программы.

С 1995 года в Borland прекратили разработку Turbo Pascal и предложили в качестве замены среду разработки Delphi, ещё более ориентированную на язык программирования Object Pascal.

Этапы.

Turbo Pascal 1.0, 1983 год. Компилирует непосредственно в машинный код. Требует 32 килобайта оперативной памяти. Стоит меньше 50 долларов. Имеет интегрированный компилятор/редактор, высокую скорость компиляции. Позволяет размещать динамические данные в куче (heap) - динамической области памяти.

Turbo Pascal 2.0, 1984 год. Увеличен размер создаваемой программы - позволяет использовать до 64 килобайт для кода, стека и данных. Версия для DOS поддерживает арифметический сопроцессор и двоично-десятичную арифметику (Binary Coded Decimal, BCD).

Turbo Pascal 3.0, 1985 год. Поддержка оверлейной структуры. Поддержка графических режимов. Специальные подпрограммы формирования изображений для IBM PC совместимых компьютеров, включая "черепаховую" графику. Инструментальные пакеты.

Turbo Pascal 4.0, 1987 год. Раздельная компиляция модулей. Размер программы ограничен только объёмом оперативной памяти. Управляемая с помощью меню интегрированная среда разработки (англ. *Integrated Development Environment, IDE*). Интеллектуальная компоновка модулей. Автономный компилятор командной строки. Убрана поддержка оверлеев. Контекстно-чувствительная система помощи.

Turbo Pascal 5.0, 1988 год. Восстановлена поддержка оверлейных структур. Встроенный отладчик. Отдельный отладчик (Turbo Debugger). Эмуляция арифметического сопроцессора. Поддержка графических драйверов BGI (Borland Graphics Interface).

Turbo Pascal 5.5, 1989 год. Объектно-ориентированное программирование. Возможность копирования в программу примеров из справочной системы. Электронный учебник на диске. Turbo Profiler - профилировщик, позволяющий оптимизировать код программы.

Turbo Pascal 6.0, 1990 год. Библиотека Turbo Vision. Новая IDE, переписанная с использованием Turbo Vision, поддерживающая мышь и редактирование нескольких файлов одновременно в разных окнах. Возможность задания во встроенном отладчике условных и по количеству проходов точек останова. Встроенный ассемблер BASM позволяющий в исходном тексте программы делать ассемблерные вставки. *Protected* поля и методы для объектов. Работающий в защищённом режиме компилятор командной строки.

Turbo Pascal for Windows. Создание 16-разрядных программ под Windows. Библиотека объектов Objects Windows Library (OWL), сходная по идеологии с Turbo Vision. Графическая IDE, работающая под Windows.

Borland Pascal 7.0, 1992 год. Выпущен Borland Pascal 7.0, включающий в себя более дешёвый и менее мощный Turbo Pascal 7.0 который поставлялся также отдельно. BP 7.0 позволял создавать программы под реальный и защищённый 16-битный режим DOS и Windows. Была введена поддержка открытых массивов, добавлено новое ключевое слово "public" для доступных полей и методов объекта. Открыты исходные тексты системных библиотек и функций времени выполнения (RTL).

*Достоинства и недостатки Турбо Паскаля.*

Достоинства.

Удобная среда разработки, включающая функциональный отладчик, доступный в любой момент.

Контекстная справочная система, по которой можно изучать язык без обращения к сторонним источникам.

Высокая скорость компиляции, высокая скорость выполнения откомпилированных программ.

Встроенная возможность использовать вставки на языке ассемблера.

Недостатки.

Компилятор рассчитан на реальный режим DOS, применение которого сходит на нет. Однако в последних версиях компилятора и среды введена поддержка защищённого режима вместе с соответствующим отладчиком (TD).

Неполная реализация объектного-ориентированного программирования.

В модуле CRT имеется ошибка (некорректный подсчёт количества циклов для функции delay, не рассчитанный на быстрые процессоры, процессоры с переменной частотой и многозадачные среды), из-за которой при запуске программы на компьютерах с тактовой частотой более 200 MHz сразу происходило аварийное завершение с сообщением "Runtime error 200 at …"

Использование файлов в Паскале.

Прежде чем перейти к обработке текстовых файлов на языке Pascal следует разобраться с понятием файл. На языке Pascal файл представляет собой упорядоченную последовательность элементов. И в отличие от общепринятого понятия файлом на языке Pascal может быть не только информация хранящиеся на дисках но и внешние устройства (такие как: дисплей, клавиатура, принтер, любой порт ЭВМ и т.д.).

Файловый тип (тип файл или файл) - это структурированный тип данных, который состоит из бесконечного (заранее неизвестного) количества элементов одного и того же типа. Элементы файла называются компонентами. Файловые типы Паскаля необходимы для того, чтобы получить возможность программного доступа к дисковым файлам. Дисковые файлы - это наборы данных на внешнем носителе.

Способы объявления файловых типов.

Общий вид объявления файлового типа

*TYPE <тип файл> = FILE [OF <тип компонент>] ;*

для текстовых файлов:

*TYPE <тип файл> = TEXT;*

для файлов записей:

*TYPE <тип файл> = FILE OF <тип запись>;*

для нетипизированных файлов:

*TYPE <тип файл> = FILE;*

Для того, чтобы использовать в программе файловые типы данных, необходимо объявить переменные файлового типа одним из следующих способов:

Способы объявления переменной файлового типа

1 способ:

*TYPE <тип файл> = FILE [OF <тип компонент>] ;*

*VAR <программный файл>: <тип файл>;*

2 способ:

*VAR <програм. файл>: FILE [OF <тип компонент>] ;*

Переменные файлового типа называют программными файлами. Файлы имеют начальную метку и конечную метку.

С каждым программным файлом связан указатель файла или текущий указатель.

Указатель файла это специальная переменная, которая хранит адрес доступной в настоящий момент компоненты файла. После выполнения операции открытия файла указатель всегда указывает на начало первой компоненты файла. Значение указателя файла изменяется в результате выполнения операций чтения файла или записи информации в файл. Если выполнена одна операция чтения или записи, то указатель файла перемещается к началу следующей компоненты.

Начало Компонента Компонента Компонента Компонента Конец

Файла 1 2 3 N файла

Рис. Файл и текущий указатель файла

Операции для работы с файлами

Каждый программный файл должен быть связан с соответствующим дисковым файлом. Такая связь организуется специальной операцией связывания:

*ASSIGN (<програм. файл>, ‘полное имя файла на диске’);*

2. Операции открытия программного файла необходима для начала работы с файлом и определения начального значения указателя файла. создаваемый и существующий файл открываются разными операциями:

для создаваемого файла:

*REWRITE (<программный файл>);*

для существующего файла:

*RESET (<программный файл>);*

3. Операции обмена информацией между операционной памятью и программными файлами:

ввод информации из файла в ОП:

*READ (<программный файл>, <переменная>);*

вывод информации:

*WRITE (<программный файл>,<переменная>);*

Операции обмена информацией с файлом организуют так называемый последовательный доступ к компонентам файла. Этот принцип аналогичен поиску информации на магнитной ленте магнитофона.

4. Прямой доступ к компоненте файла, позволяет установить указатель файла непосредственно на начало нужной компоненты. В Паскале прямой доступ возможен только по номеру компоненты. В этом случае нужно помнить о том что номер компоненты не всегда известен программисту и пользователю.

*SEEK (<программный файл>, № компоненты);*

5. Закрытие файла:

*CLOSE (<программный файл>);*

Если файл не закрыт и по какой-то причине программа завершилась аварийно, то все изменения внесенные в файл будут утеряны. Поэтому рекомендуется закрывать файл по завершении работы с ним.

6. Некоторые дополнительные операции над файлами могут очень полезны:

определяет закончился ли файл:

*EOF (<программный файл>)*

определяет размер файла в компонентах:

*FILESIZE (<программный файл>)*

определяет номер текущей компоненты:

*FILE POS (<программный файл>)*

Последовательность работы с файлами записей:

При использовании файлов записей - файлов компонентами, которых являются записи (данные комбинированного типа) нужно придерживаться следующей последовательности написания программы:

Объявить тип запись.

Объявить программный файл, компоненты которого имеют тип запись.

Объявить переменную, того же типа что и компонента файла, для обмена информацией между ОП и программными файлами.

Открыть файл существующий или создаваемый.

Для обмена информации использовать операторы ввода или вывода.

Выполнить закрытие файла.

Особенности работы с текстовыми файлами

Текстовые файлы - это файлы компонентами которых являются символьные строки. Среди текстовых файлов следует выделить стандартные текстовые файлы. Существуют особенности обработки текстовых файлов:

Стандартные текстовые файлы:

*INPUT* - стандартный файл ввода;

*OUTPUT* - стандартный файл вывода.

2. Стандартные файлы можно связать с дисковым файлом:

*ASSIGN (OUTPUT, ‘дисковый файл‘);*

3. Открытие текстового файла:

для чтения: *RESET (<программный файл>);*

для создания: *REWRITE (<программный файл>);*

для дополнения: *APPEND (<программный файл>);*

4. Обмен информацией с текстовым файлом:

чтение: *READLN (<программный файл>, переменная);*

запись: *WRITEL] (<программный файл>, переменная);*

где <переменная> должна быть объявлена как строка символов.

Работа с файлами без типа

чтение блока данных:

*BLOCKREAD (<программный файл>, переменная, n [, m]);*

запись блока данных:

*BLOCKWRITE (<программный файл>, переменная n [, m]);*

где *переменная* - переменная любого типа

*n* - размер блока,

*m* - необязательный параметр, определяющий размер блока в записях.

При чтении блока данных формируется значение *m*.

## 1. Условие и постановка задачи

Составить алгоритм и программу создания файла записей. Составить алгоритм и программу дополнения файла записями. Составить алгоритм и программу поиска информации по критерию в файле записей.

Структура записи:

Фамилия, Имя, Отчество

Дата рождения

Город

Улица

Дом

Квартира

№ паспорта

2) Внести 5-10 записей

3) Изменение записей в справочнике

4) Дополнение записей в справочнике

5) Поиск по фамилии в справочнике

## 1.1 Входная информация

Входная информация - информация, известная до решения задачи, её необходимо запросить у пользователя и ввести в оперативную память с клавиатуры из файла.

Фамилия, Имя, Отчество

Дата рождения

Город

Улица

Дом

Квартира

№ паспорта

Теперь покажем, как описывается входная программа, так как в последствии мы увидим её в тексте программы.

Для упрощения пользования ввели отдельный тип переменной - запись, она включает в себя сведения о физическом лице.

Ниже находится таблица индификаторов, в ней описана запись и соответствующие поля, которые представлены в программе.

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Назначение |
| RecBook | Запись (record) | для упрощения решения задачи |
| Поля записи | | |
| StFIO | string [20] | хранение ФИО физического лица |
| StData | string [8] | хранение даты рождения |
| StGorod | string [15] | хранение города, где проживает физическое лицо |
| StUlica | string [16] | хранение улицы |
| StDom | string [8] | хранение № дома |
| StKvartira | string [10] | хранение № квартиры |
| StNomer\_Pasporta | string [15] | хранение № паспорта |

## 1.2 Выходная информация

Выходная информация - искомый в задаче результат. Его нужно определить из входной информации и вывести пользователю на экран, печать или файл. Выходной информацией в курсовой работе являются файлы, хранящие записи с полями ФИО, дата рождения, город, улица, дом, квартира, № паспорта. Выходная информация будет заключаться в выходной информации по названию улицы. Ниже находится таблица, в которой описана выходная информация.

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Назначение |
| BookFile | файл типа RecBook | хранение данных |

## 1.3 Промежуточная информация

Промежуточная информация - промежуточные итоги решения задачи, без которых невозможно определить конечный результат.

В курсовой работе промежуточная информация служит для организации работы: для работы циклов, для считывания данных из файла и с клавиатуры.

|  |  |  |
| --- | --- | --- |
| Название переменной | Тип переменной | Назначение |
| Work | RecBook | для доступа к полям записи |
| VID | byte | для выбора вызова различных процедур |
| End\_Menu | boolean | для выхода из меню |
| Name | string [12] | хранение имени файла |
| Rez\_Find | boolean | результат поиска |
| NumRec | longint | локальная переменная принимает значение номера изменяемой записи |
| Count | integer | количество записей |

## 2. Формализация задачи

Формализация задачи - это описание задачи формальными средствами (математическими, логическими). В данной работе математические средства не использовались.

Задача может выполнять следующие действия:

создания файла записей

просмотр списка справочника

изменение записи справочника

дополнение записи в справочник

поиск физического лица по названию улицы

## 3. Алгоритм решения задачи

## 3.1 Словесный алгоритм

Алгоритм - последовательность действий, необходимых для решения задачи.

Разработка алгоритмов и программ начинается с постановки задач. На этом этапе из условия задачи определяют входную, выходную и промежуточную информацию.

На первой стадии разработки программы были описаны все переменные и их типы.

Следующим шагом стала разработка блока (процедуры) ввода исходных данных. Ввод исходных данных включает в себя создание файла для хранения данных и его заполнения. Пользователю предлагается ввести ФИО, дату рождения, город, улицу, дом, квартиру, номер паспорта. Программа может выполнять следующие действия:

*Создание нового файла*: пользователю предлагается ввести имя файла, при повторном введении старого имени - предыдущий файл заменяется новым или просто создается новый файл. Появляется приглашение к созданию нового файла, предлагается ввести число записей по всем полям записи. По завершению программа сообщает об успешном выполнении необходимого числа записей и возвращается в главное меню.

*Просмотр списка ведомости*: при выборе данного режима программа просит ввести имя файла, после чего на экран выводятся все записи данного файла.

*Изменение записи ведомости*: этот режим предназначен для редактирования некоторой записи. Сначала пользователь должен ввести имя файла и указать номер изменяемой записи. В результате на экран выводится значение полей старой записи и предлагается заполнить поля новой.

*Дополнение ведомости*: пользователь выбирает имя файла, в который необходимо внести новую запись, а затем добавляет новую запись.

*Поиск физического лица по названию улицы*: пользователь выбирает имя файла, вводит название улицы для поиска, затем на экран выводится список данных для выбранной улицы, в конце списка указывается число физических лиц с такой улицей.

Таки образом, пользователь каждый раз имеет возможность ввести необходимое имя файла. Эта особенность позволяет использовать программу при работе с множествами файлов.

Для того, чтобы создать файл, нужно ввести имя файла. С этой целью используется процедура (Name\_File).

Для добавления записей в файл используется процедура AddRec. Для обращения к полям записи используется переменная work. Пользователю предлагается ввести данные из условия задачи. Значения переменной work записываются в файл.

Для создания нового файла данных применяется процедура (Create\_Book) для работы этой процедуры переменные Ind, Count. Переменная Count содержит число записей в ведомости, а переменная Ind изменяется от 1 до значения Count. Для создания нового файла данных сначала нужно ввести имя создаваемого файла. Затем открываем новый файл для записи, используя операцию Assign, которая связывает BookFile (переменную для файла с записями) и Name (имя создаваемого файла). После этого пользователю нужно ввести количество записей в справочник. Как только создание файла данных завершится на экране появится сообщение об окончании создания файла данных. Созданный файл можно закрыть, используя операцию Close.

В программе можно вывести текущую запись на экран. Для этого применяется процедура OutputRec. При использовании этой процедуры на экране появляется таблица заполненных значений текущей записи.

Все записи можно вывести для просмотра на экран с помощью процедуры OutputAllRec. Для этого нужно открыть файл, используя операцию Reset, так как открываемый файл уже существует. Чтобы файловые операции выполнялись без ошибок применим функцию IOresult. Эта функция не имеет параметров и возвращает результат целого типа. Она работает при отключении стандартной обработки ошибок. Чтобы установить указатель на начало нужной компоненты, применим операцию Seek. Эта операция служит для того, чтобы обеспечить прямой доступ к компонентам файла. Вывод данных будет продолжаться до тех пор, пока указатель не дойдет до конца файла и не выведет все данные. Если указанного имени файла нет на диске, на экране появится надпись: "Файла с именем … на диске нет".

В созданный файл данных можно не только вносить новые записи, но и изменять их и дополнять.

Данные в файле можно изменить с помощью процедуры UpdateRec. Номер изменяемой строки обозначим через переменную NumRec. Введем значение этой переменной и установим указатель на позицию по указанному номеру записи. Введем старое значение записи (NumRec). Затем выведем запись и перейдем на следующую позицию, потом вернемся на прежнюю позицию. Зададим новое значение записи. Используя процедуру AddRec изменим запись и закроем файл.

Добавление записи осуществляется с помощью процедуры AddRecToEnd. Для этого установим текущую позицию в конец файла (Seek), введем данные в файл. На экране появится надпись: "Изменяемый файл данных имеет … записей" (FileSize). Закрываем измененный файл.

С помощью разработанной программы можно произвести поиск по названию улицы. В этой процедуре (FindFIO) используются переменные: Maska - переменная, принимающая значения заданное название улицы, Rez\_Find - переменная результата, принимает значения "False" и "True" в тех случаях, когда заданное название улицы не найдено (False) процедура будет выполняться до тех пор, пока введенное название не будет найдено (т.е. до конца записей файла). Если название улицы не найдено, на экране появится: " нет данной записи". Если улица будет найдена, на экран выведется таблица с полями записей и их значениями, внизу этой таблицы в строке будет записано количество записей для заданного пользователем улицы. Закрываем файл.

## 3.2 Блок-схема

Блок-схема основной программы:

создание

VID=1

VID=2

VID=3

VID=4

VID=5

выход

Create\_Book\_File

OutputAllRec

UpdateRec

AddRecToEnd

FindFIO

Блок-схема задания имени создаваемому файлу (Name\_File):

Name\_File

Введите имя файла

Блок-схема добавления записей в файл (Addrec):

Блок-схема процедуры создания файла (Create\_Book):

Введите номер записи

with work do

Введите ФИО, дату рождения, город, улицу, дом, квартиру, № паспорта

Write(BookFile,work)

Create\_Book\_File

Name\_File

Assign(BookFile,Name)

Rewrite(BookFile)

Ввод количества записей (Count)

Ind:=1 to count

AddRec

Close(BookFile)

Блок-схема просмотра списка справочника (OutputAllRec):

OutputAllRec

Name\_File

Assign(BookFile,Name)

Reset (BookFile)

если IOresult=0

Seek(BookFile,0)

Вывод данных из файла

While (not Eof(BookFile))

OutputRec

Файла с данным именем на диске нет

Нет Да

Блок-схема изменения записи (UpdateRec):

UpdateRec

Name\_File

Assign(BookFile,Name)

Reset (BookFile)

если IOresult=0

Файла с данным именем на диске нет

Указать имя изменяемой записи

Seek(BookFile,NumRec-1)

Задаем новое значение записи

AddRec

Старое значение записи

Close(BookFile)

Нет Да

Блок-схема дополнения справочника (AddRecToEnd):

AddRecToEnd

Name\_File

Assign(BookFile,Name)

Reset (BookFile)

если IOresult=0

Файла с данным именем на диске нет

Seek(BookFile,NumRec-1)

AddRec

Изменяемый файл данных имеет … записей

Close(BookFile)

Нет Да

Блок-схема поиска физического лица по названию улицы (FindFIO):

FindNum

Name\_File

Assign(BookFile,Name)

Reset (BookFile)

если IOresult=0

Введите улицу для поиска

Вывести таблицу

While (not Eof(BookFile))

with work do

Rez\_Find:=False

Файла с данной улицей нет

Вывести таблицу и число записей

Нет Да

Нет Да

## 3.3 Описание блок-схемы

Блок - схемы показали нам, что для решения задачи необходимы знания циклов с различными условиями (цикл с постусловием, цикл с предусловием, цикл со счетчиком). Для построения блок-схем использовались разветвляющие алгоритмы (т.е. алгоритмы, при выполнении которых порядок следования команд определяется в зависимости от результатов проверки некоторых условий).

## 4. Составление программы

Program kursach; uses crt;

Type

StFIO=string [20] ;

StData=string [8] ;

StGorod=string [15] ;

StUlica=string [16] ;

StDom=string [8] ;

StKvartira=string [10] ;

StNomer\_Pasporta=string [15] ;

RecBook=record {запись данных о физическом лице}

FIO: StFIO; {фамилия, имя, отчество}

Data: StData; {дата рождения}

Gorod: StGorod; {город}

Ulica: StUlica; {улица}

Dom: StDom; {дом}

Kvartira: StKvartira; {квартира}

Nomer\_Pasporta: StNomer\_Pasporta; {номер паспорта}

end;

Var

BookFile: file of RecBook; {переменная для файла с записями RecBook}

Work: RecBook; {переменная для доступа к полям записи}

VID: byte; {переменная для выбора вызова различных процедур}

End\_Menu: boolean; {переменная для выхода из меню}

Name: string [12] ;

Procedure Name\_File; {процедура для задания имени файла}

begin

Write ('Vvedite imya faila dannyh fizicheskogo lica: ');

Readln (Name);

end;

Procedure Addrec; {процедура для добавления записи в файл}

begin

Writeln ('Nomer fizicheskoi zapisi: ', FilePos (BookFile) +1);

with work do {переменная для обращения к полям записи}

begin

Write ('Vvedite familiu, imia, otchestvo: ');

Readln (FIO);

Write ('Vvedite Datu rogdenia: ');

Readln (Data);

Write ('Vvedite Gorod: ');

Readln (Gorod);

Write ('Vvedite Ulicu: ');

Readln (Ulica);

Write ('Vvedite Dom: ');

Readln (Dom);

Write ('Vvedite Kvartiru: ');

Readln (Kvartira);

Write ('Vvedite nomer pasporta: ');

Readln (Nomer\_Pasporta);

Write (BookFile, work); {записать в файл значение переменной work}

end;

end;

Procedure Create\_Book; {процедура для создания нового файла данных}

Var

Ind, Count: integer;

begin

Name\_File;

Assign (BookFile, Name); {открыть файл для записи}

Rewrite (BookFile);

Writeln ('Sozdanie zapisei faila: ‘Name);

Write ('Vvedite chislo zapisei v spravochnike: ');

Readln (Count);

For Ind: =1 to count do

AddRec;

Writeln ('Sozdanie faila dannyh fizicheskogo lica zaversheno');

Writeln ('Fail dannyh imeet ', FileSize (BookFile),' zapisei');

Close (BookFile);

end;

Procedure OutputRec; {процедура вывода текущей записи на экран}

begin

Read (BookFile, work);

with work do

begin

Writeln ('------------------------------------------------------------------------------');

Write ('| ', FilePos (BookFile),' | ', FIO,' | ', Data,' | ‘, Gorod,' | ', Ulica,' | ', Dom,' | ');

Writeln (Kvartira,' | ', Nomer\_Pasporta,' |');

Writeln ('------------------------------------------------------------------------------');

end;

end;

Procedure OutputAllRec; {процедура вывода всех записей файла на экран}

begin

Name\_File;

Assign (BookFile, Name);

{$I-} {отключение стандартной обработки ошибок}

Reset (BookFile); {поиск открытого файла}

{$I+} { включение стандартной обработки ошибок}

If IOresult=0 then

begin

Seek (BookFile, 0); {установка на первую запись}

Writeln ('\*\*\*Vyvod dannih fizicheskogo lica iz faila ', Name,' \*\*\*');

Writeln ('------------------------------------------------------------------------------');

Writeln ('| N | FIO | Data rogdenia | Gorod | Ulica | Dom | Kvartira | Nomer Pasporta |');

Writeln ('------------------------------------------------------------------------------');

While (not Eof (BookFile)) do OutputRec; {условие окончания цикла}

End

Else Writeln ('Faila s imenem ', Name,' na diske net');

end;

Procedure UpdateRec; {изменение записи файла}

Var

NumRec: longint; { локальная переменная принимает значение номера изменяемой записи}

begin

Name\_File;

Assign (BookFile, Name);

{$I-}

Reset (BookFile);

{$I+}

If IOresult=0 then

begin

Write ('Ukazat nomer izmenayemoi zapisi: ');

Readln (NumRec);

Seek (BookFile, NumRec-1); {установка файловой позиции по указанному номеру записи}

Writeln ('staroe znachenie zapisi: ‘ NumRec);

OutputRec;

Seek (BookFile, NumRec-1); {возврат на прежнюю позицию}

Writeln ('Zadaem novoe znachenie ', NumRec,' zapisi');

AddRec;

Close (BookFile);

end

Else Writeln ('Faila s imenem ', Name,' na diske net');

end;

Procedure AddRecToEnd; { процедура дополнения записи в конец файла}

begin

Name\_File;

Assign (BookFile, Name);

{$I-}

Reset (BookFile);

{$I+}

If IOresult=0 then

begin

Seek (BookFile, FileSize (BookFile)); {установка текущей позиции в конец файла}

AddRec;

Writeln ('Izmenennyi fail dannyh imeet ', FileSize (BookFile),' zapisei');

Close (BookFile);

end

Else Writeln ('Faila s imenem ', Name,' na diske net');

end;

Procedure FindFIO; {процедура поиска названия улицы физического лица}

Var

Maska: StUlica; {хранит название улицы искомого лица}

Rez\_Find: boolean; {результат поиска}

CountRec: integer; {переменная, значение которой равно числу записей с такой улицей}

begin

Name\_File;

Assign (BookFile, Name);

{$I-}

Reset (BookFile);

{$I+}

If IOresult=0 then

begin

Write ('Vvedite ulicu dla poiska: ');

Readln (Maska);

Rez\_Find: =False; {улица не найдена}

CountRec: =0;

Writeln ('------------------------------------------------------------------------------');

Writeln ('| N | FIO | Data rogdenia | Gorod | Ulica | Dom | Kvartira | Nomer pasporta |');

Writeln ('------------------------------------------------------------------------------');

While (not Eof (BookFile)) do {просмотреть все записи до конца файлов}

begin

Read (BookFile,work);

with work do

If Pos (Maska,Ulica) <>0 then {найдена запись физического лица}

Begin

Rez\_Find: =True;

Inc (CountRec);

Writeln ('------------------------------------------------------------------------------');

Write ('| ',FilePos (BookFile),' | ',FIO,' | ',Data,' | ',Gorod,' | ',Ulica,' | ',Dom,' | ');

Writeln (Kvartira,' | ',Nomer\_Pasporta,' |');

Writeln ('------------------------------------------------------------------------------');

end;

end;

If Rez\_Find then

Writeln ('Chislo zapisei dlya ulic ',Maska,' = ',CountRec)

Else Writeln ('V spravochnike net takoi ulici ',Maska);

Close (BookFile);

end

Else Writeln ('Faila s imenem ',Name,' na diske net');

end;

begin

clrscr;

End\_Menu: =False;

Repeat

Writeln ('\*\*\*Spravochnic dannih fizicheskih lic\*\*\*');

Writeln ('Vyberite vid raboty');

Writeln ('1 - sozdanie novogo faila');

Writeln ('2 - prosmotr spiska spravochnika');

Writeln ('3 - izmenenie zapisi');

Writeln ('4 - dopolnenie spravochnika');

Writeln ('5 - poisk po familii');

Writeln ('0 - zavershenie raboty');

Write ('Vash bybor: ');

Readln (VID);

case VID of {вызов разных процедур в зависимости от видов работы}

1: Create\_Book;

2: OutputAllRec;

3: UpdateRec;

4: AddRecToEnd;

5: FindFIO;

0: End\_menu: =True;

end;

Writeln ('Dlya prodolzheniya nazhmite Enter');

Readln;

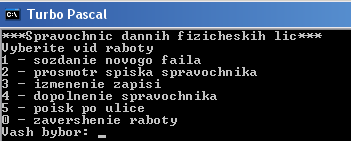
clrscr;

Until End\_Menu;

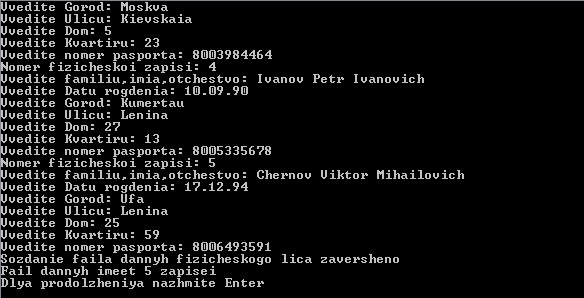
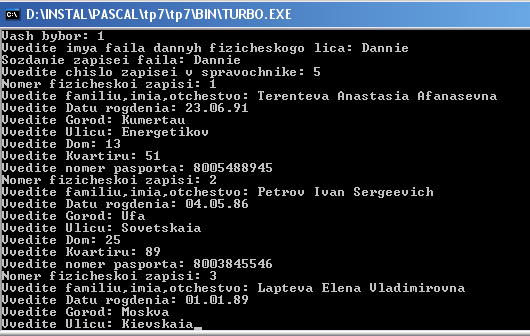
end.

## 5. Контрольный пример

Наберем данную программу на языке Паскаль. Откомпилируем ее. На экране появится меню задач для файла:



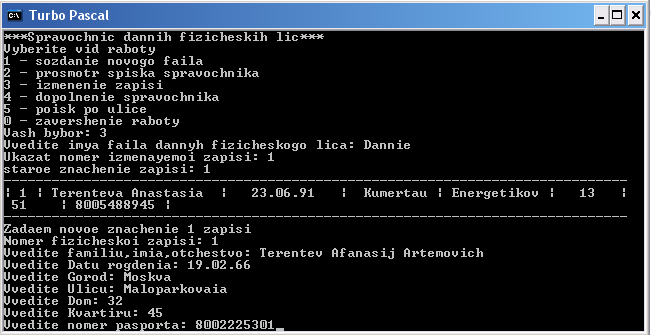
При выборе "1" создадим новый файл под именем "\*\*\*Spravochnik dannih fizicheskih lic\*\*\*":



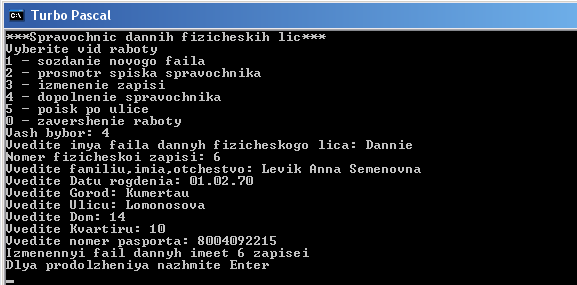
При выборе "2" можно просмотреть весь созданный список:



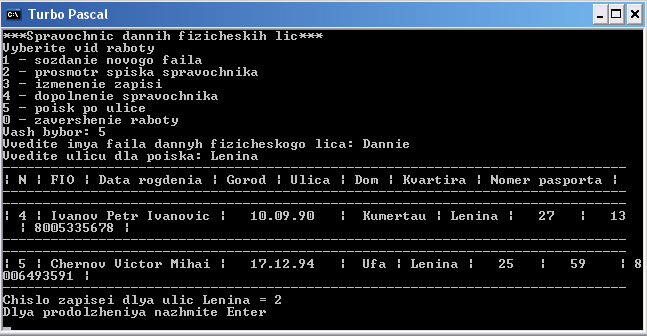
При выборе "3" изменим одну из записей файла:



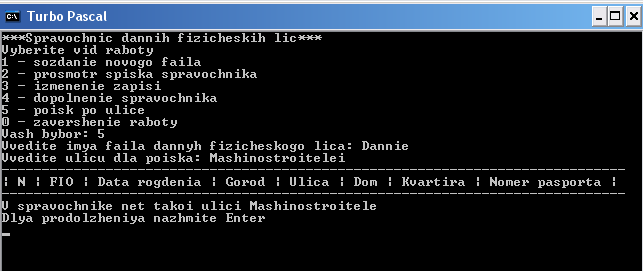
При выборе "4" дополним справочник еще одной записью:



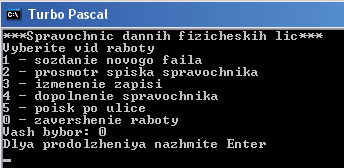
При выборе "5" пользователю будет предложено ввести название улицы, введем название (Ленина) и получим данные о физических лиц с такой улицей:



В случае, если заданной улицы нет нет, то на экране появится:



При выборе "0" программа закончит свою работу:



## Заключение

Задача, которая поставлена в курсовой работе была выполнена. Мы показали, что задача может выполнять несколько действий:

Создание файла

Заполнение файла данными, содержащие поля:

Фамилия, имя, отчество

Дата рождения

Город

Улица

Дом

Квартира

Номер паспорта

Просмотр всех записей файла или текущей записи

Изменение записи, созданной в файле

Дополнение записи в файл

Поиск записи по названию улицы.

Все эти действия были выполнены с помощью процедур.

Процедура - это часть программы, в которой выполняется специфическое действие. Процедуры в Турбо Паскале могут быть внутренними и внешними.

Внутренние процедуры описываются в разделе описания процедур основной программы. Внешние процедуры оформляются в виде автономно компилируемых модулей.

*Формат описания процедур*:

PROCEDURE <имя проц> (список форм. Парам);

Разделы описания

BEGIN

Раздел операторов

END.

*Имя проц*. - имя процедуры, любое имя задаваемой программистом, обычно соответствующим назначению процедуры, состоящее из латинских букв и цифр 0-9.

## Список литературы

1. Грачева Л.Н. - Программирование в среде Turbo-Pascal 7.0: Лабораторный практикум по дисциплине: "Программирование и основы алгоритмизации". - Уфа: РИО РУНМЦ МО РБ, 2008.
2. Молодцов В.А. - Репетитор по информатике для подготовки к экзаменам. - Ростов н/Д: Феникс, 2007.
3. Острейковский В.А. - Информатика: Учеб. для вузов. - М.: Высш. шк., 1999.
4. Попов В.Б. - Паскаль и Дельфи. Учебный курс. - СПб.: Питер, 2005.