Министерство образования и науки Украины

Севастопольский государственный технический университет

Кафедра Информационных систем

Пояснительная записка

к курсовому проекту

Разработка программы обработки экономической информации

по курсу Основы программирования и алгоритмические языки

Выполнил:

ст. гр. И-21д

Климашевский С.В.

Проверил:

Руководитель проекта

Крицкий А.В.

Севастополь

2003

**Аннотация**

В данном документе описывается программа, написанная в соответствии с постановкой задачи на курсовое проектирование по теме "Обработка экономической информации" по дисциплине "Основы программирования и алгоритмические языки". Данная программа предназначена для удобной обработки информации, содержащей сведения о студентах, код группы, фамилия студента, количество пропущенных занятий и число оправданных пропусков, составления и вывода на экран и(или) в файл отчётных таблиц. Входными данными является таблица, записью которой являются код группы, фамилия студента, количество пропущенных занятий, количество оправданных пропусков, количество не оправданных пропусков, сумма не оправданных пропусков и процентное соотношение пропусков. Для проверки работы программы разработан тестовый пример. Результаты тестирования доказывают, что программа правильно выполняет все операции по обработке входных данных и формирования выходных данных.

**Содержание**

Введение

1. Назначение и область применения

2. Технические характеристики

2.1 Постановка задачи

2.2 Описание основных типов данных

2.3 Входные данные

2.4 Выходные данные

2.5 Внутреннее представление данных

2.6 Описание алгоритма программ

3. Технико-экономические показатели

4. Тестовый пример

Заключение

Библиографический список

Приложение

**Введение**

По окончанию курса “Основы программирования и алгоритмические языки” стоит задача в закреплении навыков программирования на ЭВМ. Она состоит в написании программы на языке высокого уровня PASCAL или Си знакомстве с организацией и принципами действия операционных систем, изучение правил оформление программной документации.

В современном мире обработки информации очень удобно использование диалогового режима работы. Программа была разработана в среде Turbo PASCAL 6.0 .

**1. Назначение и область применения**

Данная программа предназначена для удобной обработки информации, содержащей сведения о студентах, код группы, фамилия студента, количество пропущенных занятий и число оправданных пропусков, составления и вывода на экран и в файл отчётных таблиц, задаваемый пользователем.

Область применения программы – различные учебные учреждения, где ведется учет пропусков занятий студентами .

Требованием к аппаратному и программному обеспечению является наличие любого IBM-совместимого компьютера с наличием накопителя на гибких и (или) жестких магнитных дисках, видеоадаптера класса VGA и наличием достаточного количества оперативной памяти . На компьютере должна быть установлена операционная система MS-DOS v2.1(или выше) или любая другая, совместимая с ней. Применяться может любым пользователем, который хочет проверитьпосещаемость для какой-либо группы лиц.

**2. Технические характеристики**

**2.1 Постановка задачи**

Требуется написать программу обработки экономической информации, осуществляющую ввод исходной информации, обработку в соответствии с вариантом задания №8 методических указаний, а также печать содержимого исходных файлов и файлов расчетов. Выводимую информацию оформить в виде таблицы со всеми необходимыми полями записей и результатов расчетов.

Даны сведения за месяц о пропуске занятий студентами групп. Структура записи: шифр группы (6 символов), фамилия (15 символов), пропущено часов, оправдано часов. Подсчитать количество неоправданных часов по каждому студенту, суммарные показатели по каждому виду, а также процент неоправданных пропусков в целом.

Работу программы организовать в форме меню:

1). Загрузка информации из исходного файла

2). Обработка данных таблицы

3). Удаление данных о каком-либо студенте.

4). Добавление данных о каком-либо новом студенте.

5). Поиск элемента в таблице

6). Просмотр данных

7). Запись данных в новый файл

8). Выход.

Программа реализована на языке программирования высокого уровня Pascal. В основе программы лежит двунаправленный список с пятью информационными полями.

Мною был выбран двунаправленный список, потому что он прост в управлении, доступ к элементам списка осуществляется путем указания имени структуры и соответствующего информационного поля, содержимую информацию,в двунвправленном списке, можно вывести так, что после вывода будет осуществлять просмотр данных с использованием некоторы клавиш: -вверх- и -вниз-. Использование двунаправленных списков позволяет работать с динамической памятью, что позволяет экономнее использовать память и ресурсы компьютера. Программа сохраняет выходные данные в файле на жестком диске.Выходнй файл имеет такой формат, который позволяет загружать (при новом запуске программы ) из него.

Просмотр осуществляется дважды: 1-ый раз – при загрузке данных из исходного файла , 2-ой раз - при просмотре обработанных данных (именно они будут записаны в выходной файл).

**2.2 Описание основных типов данных**

Для внутреннего хранения входных данных о слогах используется комбинированный тип данных:

type

zap=record

fio:string[15];

nomergrupp:string[4];

propusk:integer;

opravdan:integer;

neopravdan:integer;

summa:integer;

procent:real;

end;

Информация,содержащаяся в полях:

fio: массив для хранения фамилии студента;

nomergrupp: шифр группы;

propusk:число пропусков занятий студентом;

opravdan:число пропусков занатий по уважительной причине;

neopravdan: число пропусков занатий без уважительной причины;

summa:общая сумма пропусков по всем студентам;

procent:процент пропусков без уважительной причины;

Для того чтобы возможно было обрабатывать большое число записей из исходной ведомости, необходимо организовать их структуру данных. Для удобной работы с записями в качестве структуры данных была выбрана линейная структура - двунаправленный список

. При использовании двунаправленного списка, возможно организовать просмотр записей всего списка: от начала в конец (прямое направление) и из конца в начало (обратное), что используется в процедуре просмотра при перемещении от одной записи к другой.

**2.3 Входные данные**

Входные данные передаются в программу следующим способом:

Чтение входных данных из предварительно подготовленного файла, определенного формата.

Предполагается, что входные данные содержатся в текстовом файле правильного формата, имя которого dano.txt, а местонахождение - текущий каталог.

Пример входных данных текстовый файл приведен на рисунке 2.1

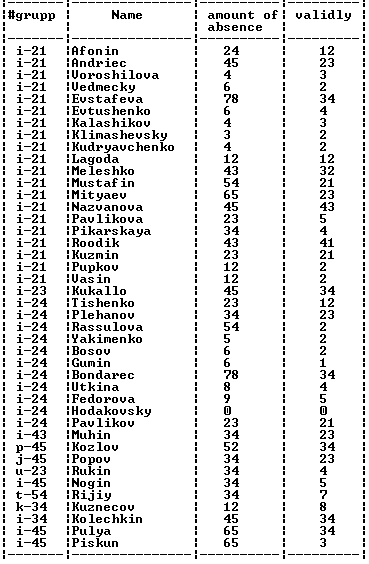


Рисунок 2.3.1 – Пример файл входных данных

**2.4 Выходные данные**

В процессе ввода входных данных из файла, формируется динамическая структура – двунаправленный список. Информационным полем элемента списка является запись таблицы входных данных. Для просмотра данных используются функции обхода двунаправленного списка.

В результате обработки данных составляется ведомость, которая при выборе меню watch and write output file отображается на экране и записывается в новый файл с выполненными по заданию расчетами в виде таблицы:

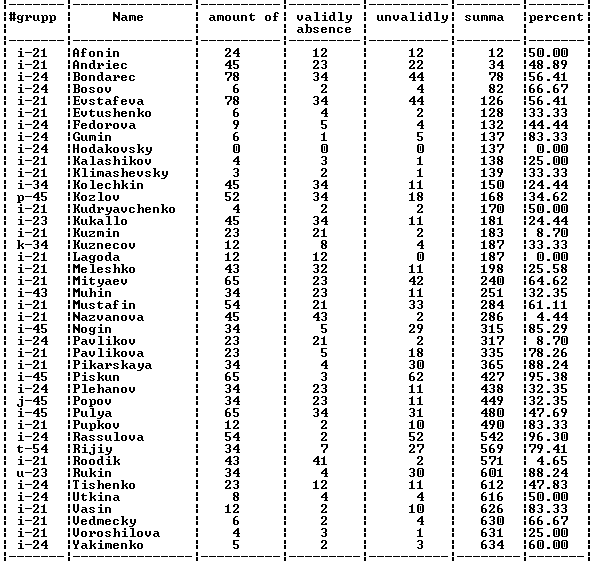


Рисунок 2.4.1 – Пример файла выходных данных

**2.5 Внутреннее представление данных**

Программа написана на языке высокого уровня Turbo Pascal и скомпилирована компилятором Borland Pascal 7.0 .

Файл, может включать в себя неизвестное заранее количество строк, но должен иметь определённый формат, поэтому для хранения входной информации статические переменные неприемлемы. Наиболее подходящим оказывается двунаправленный список , так как в нем возможны: перестановка порядка следования элементов без копирования самой структуры; поиск как в прямом, так и обратном направлении; удаление элементов наиболее оптимальным образом. В тоже время не требуется поддержка сложных структур данных и методов работы, как в деревьях.Но тем не менее скорость работы двунаправленного списка ниже , чем у нелинейных структур, например, деревьев.

**2.6 Описание алгоритма программ**

Структура алгоритма задачи для процедуры org;- считывание из файла и создание двунаправленного списка с упорядоченной структурой по полю fio (фамилия студента) по алфавиту. (рис. 2.6.1.) .

Блок1-обнуляет число элементов.

Блок2-чтение имени файла.

Блок3-проверка введено ли имя.

Блок4-Вывод окна сообщений о том , что имя не введено.

Блок5- присвоение булевой переменной определяющей есть ли данный файл значения false.

Блок6-выход из процедуры.

Блок7-открытие файла.

Блок8-проверка на наличие файла.

Блок9-вывод на экран сообщения о том . что файла такого нет.

Блок10-булевой переменной определяющей есть ли данный файл значения true.Обнуление числа строк

Блок11- присвоение булевой переменной определяющей есть ли данный файл значения false.

Блок12-цикл пока не конец файла

Блок13-цикл пока не почитал 4 строки

Блок15-подсчёт строк

Блок16-чтение строки

Блок17-считывание частей строк в промежуточные переменные.

Блок18-проверка двузначное ли число пропусков

Блок19-(да)копируем из строки 1 эл-т

Блок20-(нет)-2 эл-та

Блок21-перевод строки в целое число

Блок22-проверка двузначное ли число пргопусков по ув. причине

Блок23-(да)копируем из строки 1 эл-т

Блок24-(нет)-2 эл-та

Блок25-перевод строки в целое число

Блок26-выделние памяти и присвоение всех указателей для первого элемента.

Блок27-цикл пока не конец файла

Блок28-чтение строки

Блок29-считывание частей строк в промежуточные переменные.

Блок30-проверка двузначное ли число пропусков

Блок31-(да)копируем из строки 1 эл-т

Блок32-(нет)-2 эл-та

Блок33-перевод строки в целое число

Блок34-проверка двузначное ли число пргопусков по ув. причине

Блок35-(да)копируем из строки 1 эл-т

Блок36-(нет)-2 эл-та

Блок37-перевод строки в целое число

Блок38-присвоение указателю значение первого

Блок39-цикл пока не конец списка

Блок40-сравнение считанной переменной с фамилиями списка

Блок41-(больше)-перевод указателя на следующий эл-т

Блок42-выход из цикла

Блок43-выделение памяти новому эл-ту

Блок44-прверка, если указатель на последний эл-т

Блок45-(да)добавление эл-та в конец

Блок46-(нет)проверка указатель на первый или нет

Блок47-(да)добавление эл-та в начало

Блок48-добавление в любле другое место

Блок49-указателю конца приваем значение nil

Блок50-закрытие файла

Блок51-конец.



Рисунок 2.6.1-процедура чтения информации из файла и организация списка.



Рисунок 2.6.1 – продолжение



Рисунок 2.6.1- продолжение



Рисунок 2.6.1 –продолжение.



Рисунок 2.6.1 – продолжение.

Структура алгоритма задачи для процедуры zapis\_v\_fail; - сохранение результата в файл (рис. 2.6.2)

Блок1-чтение имени файла

Блок2-установка связи с файлом

Блок3-открытие его в режиме записи

Блок4-присвение первого значения указателю

Блок5-запись шапки таблицы в файл

Блок6-цикл пока не конец списка -

Блок7-(конец) запись в файл нижней границы таблицы

Блок8-очистка экрана

Блок9-вывод на экран информации о том, что файл был записан

Блок10-закрытие файла

Блок11-конец

Блок12запись данных в файл

Блок13-первод указателя на следующий элемент



Рисунок 2.6.2-процедура записи данных в файл.



Рисунок 2.6.2-продолжение

Структура алгоритма задачи процедуры okno(t:integer) .Печать меню и выделение строки под номером t. (Рис.2.6.3);

Блок1-присваение значения номера строки переменной p

Блок2-печать верхней рамки меню

Блок3-проверка значения p=1

Блок4-(да)цвет фона строки изменяется на чёрный

Блок5-(нет)печать 1-ого пункта меню

Блок6-возврат к прежнему фону

Блок7-проверка значения p=2

Блок8-(да)цвет фона строки изменяется на чёрный

Блок9-(нет)печать 2-ого пункта меню

Блок10-возврат к прежнему фону

Блок11-проверка значения p=3

Блок12-(да)цвет фона строки изменяется на чёрный

Блок13-(нет)печать 3-ого пункта меню

Блок14-возврат к прежнему фону

Блок15-проверка значения p=4

Блок16-(да)цвет фона строки изменяется на чёрный

Блок17-(нет)печать 4-ого пункта меню

Блок18-возврат к прежнему фону

Блок19-проверка значения p=5

Блок20-(да)цвет фона строки изменяется на чёрный

Блок21-(нет)печать 5-ого пункта меню

Блок22-возврат к прежнему фону

Блок23-проверка значения p=6

Блок24-(да)цвет фона строки изменяется на чёрный

Блок25-(нет)печать 6-ого пункта меню

Блок26-возврат к прежнему фону

Блок27-проверка значения p=7

Блок28-(да)цвет фона строки изменяется на чёрный

Блок29-(нет)печать 7-ого пункта меню

Блок30-возврат к прежнему фону

Блок31-проверка значения p=8

Блок32-(да)цвет фона строки изменяется на чёрный

Блок33-(нет)печать 8-ого пункта меню

Блок34-возврат к прежнему фону

Блок35-печать нижней рамки меню

Блок36- возврат к прежнему фону

Блок37-конец.



Рис 2.6.3.процедура okno-печать меню.



Рис 2.6.3.-продолжение



Рисунок 2.6.3-продолжение

Структура алгоритма задачи процедуры poisk(x:ukaz;imya:string) ищет необходимого студента в таблице.(Рис.2.6.4) .

Блок1-обнуление переменных числа выводимых на экран строк(n), числа элементов с требуемым именем(i) и ррисвоение указателю значения первого.

Блок2-цикл пока не конец списка

Блок3-подсчёт числа э-тов всписке с данным именем

Блок4-первод указателя на следующий эл-т

Блок5-проверка равно ли число студентов с аким именм 0

Блок6-(да)вывод на экран сообщения о том , что такого э-та нет

Блок7-(нет)изменение указателя на переданный

Блок8-выход из процедуры

Блок9-печать шапки таблицы

Блок10-цикл пока число n<>0 (число выводимых строк)

Блок11-подсчёт числа таких эл-тов

Блок12-проверка тот ли это эл-т по фамилии

Блок13-(не тот)не выделять

Блок14-(тот)выделить

Блок15-печать эл-та

Блок16-возвращение к прежнему цвету

Блок17изменение указателя на следующий

Блок18-печать нижней границы таблицы

Блок19-конец.



Рис.2.6.4-процедура поиска нужного студента.



Рис.2.6.4-продолжение.

Структура алгоритма задачи для структурных схем , выполненных на чертеже формата А1.

Для основной программы:

Блок1-присваивает значению номера строки еденицу

Блок2-организовывает печать меню с выделенной первой строкой

Блок3-организует бесконечный цикл

Блок4-организует считывание клавиш

Блок5-сравнивает код клавиш с кодом enter

Блок6-сравнивает код клавиши с 0

Блок7-роганизовывает вызов процедур по номеру строки

Блок8-чтение клавиши

Блок9-сравнивает код клавиши с кодом стрелки вниз

Блок10-привыполнении условия в блоке 9 увеличивает номер текущей строки на 1

Блок11-сравнивает номер строки с его максимальным значением

Блок12-сравнивает код клавиши с кодом стрелки вверх

Блок13-при выполнении условия в блоке 12 уменьшает значение текущей строки на 1

Блок14- сравнивает номер строки с его минимальным значением

Блок15-в случае выполнения условия блока 14 присваивают номеру строки максимальное значение

Блоки16,17,18-организуют вызов процедуры печати меню

Блоки19-25- организуют вызов процедур, соответствующих номеру строки

Блок26-выход из программы.

Блок27- при выполнении условия в блоке 11 присваивает значению номера текущей строки минимальное

Блок28-вызывает процедуру печати меню.

Для процедуры obrabotka:

Блок1-присвамивает указателю начальное значение и обнуляет промежуточную величину x.

Блок2-создает цикл обхода списка.

Блок3 –находит значения поля неоправданных пропусков для каждого эл-та.

Блок4-прверяет не равно ли число пропусков 0.Тогда:

Блок5-если не равен находит процент пропусков без уважительной причины.

Блок6- присваивает значение 0 полю процента пропусков без уважительной причины.

Блок7-нахождение промежуточной величины x для дальнейшего нахождения значения поля суммы общих прогулов без уважительной причины.

Блок8-нахождение значения суммы общих пропусков без уважительной причины. и перевод указателя на следующий эл-т.

Блок9-выводит на экран информацию о том , что обработка данных завершилась.

Описание процедур.

procedure org;-организует ввод данных из файла, создание двунаправленного сриска упорядоченного по фамилиям студентов по алфавиту.

procedure okno(t:integer);-выводит на экран меню программы.

procedure dob;-добавляет нового студента в таблицу ,не нарушая упорядоченности.

procedure udal;-удаление элемента.

procedure poisk;-поиск студента по фамилии.

procedure obrabotka;-обработка данных по заданию.

procedure zapis\_v\_fail;-сохранение текущей таблицы данных в файл, который укажет пользователь.

procedure klavishi(nom:integer);-считывает коды клавишь и позволяет просматривать всю таблицу в процедурах поиска и просмотра.

procedure prosmotr\_2(x:ukaz);-выводит на экран текущую таблицу со всеми изменениями.

Выбор языка программирования.

Язык Turbo Pascal и компилятор выбраны из-за его гибкости, широкого распространения (субъективный фактор), малого объема исполняемого файла и достаточно высокой скорости выполнения. Компилятор Borland Pascal 7.0 . к тому же включает в себя опциональную возможность оптимизации кода программы как по скорости, так и по размеру. Основным фактором выбора языка Pascal была его более простая

работа со строками. А в данном курсовом пректе это очень важно так как чтение исходных данных производится из текстового файла , данные в котором хранятся в виде таблицы. Этот язык программирования относительно прост, он гибкий и развитый в отношнии типов языка .Язык Си более сложный при написании , поэтому в данном проекте задача была реализована на я зыке Turbo Pascal , хотя не отрицается возможнсть успешной реализации задачи на языке Си.

**3. Технико-экономические показатели**

В результате компиляции программы компилятором Borland Pascal 7.0 . был получен исполняемый файл формата DOS EXE размером 20 Кбайт, что является приемлемым как для компьютеров десятилетней давности, так тем более, современных.

Для работы программа требует приблизительно 200 Кбайт оперативной памяти (зависит от размера входного файла). Данный объем позволяет программе выполнятся на самом широком круге компьютеров, фактически всех поддерживающих операционную систему DOS.

Для выполнения требуется процессор класса 80x86, цветной монитор, поддерживающий текстовый режим 80х50.

Размер входного файла зависит от количества символов в нем и для 44 элементов (студентов) составляет 2Кбайта. Размер выходного файла зависит от преобразований.

В программе отсутствует ресурсоемкие процедуры, поэтому производительность центрального процессора не критична.

Программа может работать под управлением операционной системы класса DOS, при этои сохраняет работоспособность и в режиме эмуляции MS-DOS операционной системой Windows9X,Me и RedHat Linux. Работоспособность в режиме эмуляции под другими операционными системами не проверялось, хотя эта возможность не исключается.

Таким образом программа может выполняться практически на любом доступном PC компьютере.

**4. Тестовый пример**

Проведем эксперимент, чтобы убедиться в работоспособности программы.

Возьмем исходную таблицу(рис.4.1):

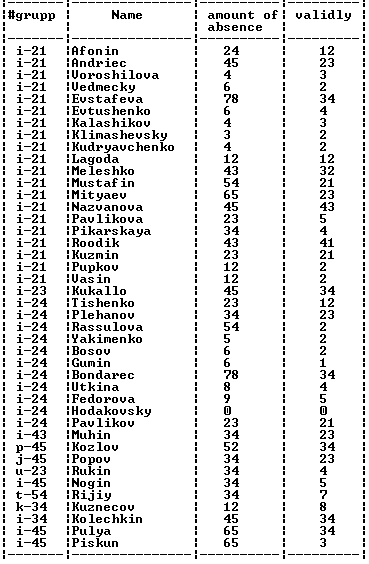
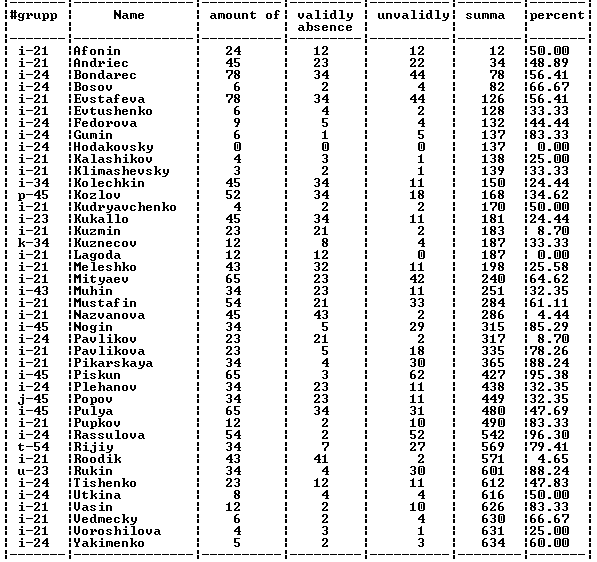


Рис.4.1- исходная таблица

Результат после обработки



Эти данные также будут сохранены в файл с указанным именем и путём к нему.

**Заключение**

Данная программа, составлена в соответствии с постановкой задачи на курсовое проектирование по теме "Разработка программы обработки информации" по дисциплине "Основы программирования и алгоритмические языки" (ОПиАЯ). При написании программы использованы методические указания по курсовому проектированию по дисциплине "ОПиАЯ". Для проверки работоспособности программы и правильности обработки входных данных разработан тестовый пример. Тестирование программы подтвердило, что программа правильно выполняет обработку данных и выдаёт верные результаты. Всё это свидетельствует о работоспособности программы и позволяет сделать вывод о пригодности программы к решению практических задач по обработке экономической информации.

**Библиографический список**

1. Немнюгин С.А. –Turbo Pascal учебник.”Питер”,2001.-496л

2. Фараонов В.В Turbo Pascal 7.0.”Нолидж”2001.

3. Конспект по лекциям ОП и АЯ.

**Приложение**

Листинг программы

*program Kursovik;*

*Uses crt;*

*type*

*zap=record*

*fio:string[15];*

*nomergrupp:string[4]; //шифр группы*

*propusk:integer; //число пропусков*

*opravdan:integer; // оправданных*

*neopravdan:integer; //неоправданных*

*summa:integer; //общая сумма неоправданных*

*procent:real; //процент неоправданных*

*end;*

*ukaz=^querty;//создаём структуру*

*querty=record*

*inf:zap;*

*nr,nl:ukaz;*

*end;*

*var*

*sozdan:boolean; //проверяет есть ли файл*

*fp,ft,ff:text;*

*verh,newE,right,left,temp:ukaz;*

*z:zap;*

*c:char;*

*x,kolvo,otpech,i,number:integer;//номер строки (для меню),кол-во эл-тов*

*procedure okno(t:integer);FORWARD;//печать меню*

*procedure zapis\_v\_fail;FORWARD;//запись в файл*

*{-----------------ORGANIZACIYA IZ FAILA------------------------------------}*

*procedure org;//организация из файла двунаправленного списка по алфавиту*

*var*

*pop:ukaz;*

*d,stroka,t1,t2,t3,t4:string;*

*p3,p4:array[1..3]of char;*

*schet,pole3,pole4,errcode:integer;*

*begin*

*kolvo:=0;*

*gotoxy(25,25);*

*textcolor(black);*

*writeln('Input the name of file');*

*textcolor(white);*

*gotoxy(27,27);*

*readln(d);*

*if d='' then*

*begin*

*clrscr;*

*gotoxy(25,25);*

*writeln('This file does not exists!!!!!!!');*

*gotoxy(25,26);*

*writeln('Push the -enter- and input name of file again. ');*

*readln;*

*sozdan:=false;*

*exit;*

*end;*

*assign(ft,d);*

*{$I-}*

*reset(ft);*

*if IOResult <>0//проверка есть ли файл*

*then*

*begin*

*clrscr;*

*gotoxy(25,25);*

*writeln('This file does not exists!!!!!!!');//сообщение , что файла нет*

*gotoxy(25,26);*

*writeln('Push the -enter- and input name of file again. ');//ещё раз надо ввести имя файла*

*readln;*

*sozdan:=false;//переменной создания файла присвоить ложь*

*exit;*

*end;*

*sozdan:=true; //переменной создания файла присвоить правда*

*schet:=0;*

*while not EOF(ft) do//цикл пока не конец файла*

*begin*

*while schet<>4 do//пропуск 4 строк*

*begin*

*readln(ft,stroka);*

*schet:=schet+1;*

*end;*

*readln(ft,stroka);//считывание строки*

*t1:=copy(stroka,3,7);*

*t2:=copy(stroka,10,15);*

*{perevod stroki v integer}*

*if stroka[30]= ' '*

*then*

*t3:=copy(stroka,29,1)*

*else*

*t3:=copy(stroka,29,2);*

*val(t3,pole3,errcode);*

*{perevod stroki v integer}*

*if stroka[42]= ' 'then*

*t4:=copy(stroka,41,1)*

*else*

*t4:=copy(stroka,41,2);*

*val(t4,pole4,errcode);*

*new(newE);..//создание нового эл-та*

*newE^.inf.fio:=t2;//идёт присвоение по полям*

*newE^.inf.nomergrupp:=t1;*

*newE^.inf.propusk:=pole3;*

*newE^.inf.opravdan:=pole4;*

*newE^.nr:=nil;*

*newE^.nl:=nil;*

*left:=newE;Right:=newE;*

*verh:=newE;*

*kolvo:=kolvo+1;*

*{--sozdali 1-iy element}*

*while Not eof(ft) do//покане конец файла*

*begin*

*readln(ft,stroka);//чтение строки*

*if stroka[3]='-'then //сколько знаков не конец ли таблицы*

*exit;*

*t1:=copy(stroka,3,7); //копирывание*

*t2:=copy(stroka,10,15);*

*{perevod stroki v integer}*

*if stroka[30]= ' 'then //сколько знаков не конец ли таблицы*

*t3:=copy(stroka,29,1)*

*else*

*t3:=copy(stroka,29,2); //копирывание*

*val(t3,pole3,errcode); //чтение строки в инт*

*{perevod stroki v integer}*

*if stroka[42]= ' 'then*

*t4:=copy(stroka,41,1)*

*else*

*t4:=copy(stroka,41,2);*

*val(t4,pole4,errcode);*

*{-----sozdanie o4eredi-------------------------------------}*

*pop:=verh; // указатель на на чало*

*while pop<>nil do// цикл пока неконец списка*

*if t2>= pop^.inf.fio//сравнение по фамилиям*

*then*

*pop:=pop^.nr//перевод указателя*

*else*

*break; //прервать цикл*

*new(newE); //создание нового элемента*

*newE^.inf.fio:=t2;*

*newE^.inf.nomergrupp:=t1;*

*newE^.inf.propusk:=pole3;*

*newE^.inf.opravdan:=pole4;*

*if pop=nil then//добавление в конец нового элемента*

*begin*

*right^.nr:=newE;*

*newE^.nl:=right;*

*newE^.nr:=nil;*

*right:=newE;*

*kolvo:=kolvo+1;*

*end*

*else*

*if pop=verh then//добавление в начало*

*begin*

*pop^.nl:=newE;*

*newE^.nr:=pop;*

*newE^.nl:=nil;*

*kolvo:=kolvo+1;*

*verh:=newE;*

*left:=newE;*

*end*

*else*

*begin//добавление в сердину*

*pop^.nl^.nr:=newE;*

*newE^.nl:=pop^.nl;*

*pop^.nl:=newE;*

*newE^.nr:=pop;*

*kolvo:=kolvo+1;*

*end;*

*end;*

*end;*

*right^.nr:=nil; // правый указатель на конкц*

*close(ft);exit// закрытие файла*

*end;*

*{------------------------------------------------------------------------}*

*{------------------prosmotr\_2-----------------------------------------}*

*{------------------------------------------------------------------------}*

*procedure prosmotr\_2(x:ukaz);*

*var*

*n:integer;*

*c:char;*

*begin*

*clrscr;*

*n:=0; //число выводимых строк*

*newE:=x; //указатель*

*writeln('|------|---------------|----------|---------|-----------|--------|-------|');*

*writeln('|#grupp| Name | amount of| validly | unvalidly | summa |percent|');*

*writeln('| | | | absence | | | |');*

*writeln('|------|---------------|----------|---------|-----------|--------|-------|');*

*while (newE<>right^.nr) and (n<=6) do//цикл пока не конец списка и число строк =6*

*begin*

*inc(n);*

*write('|', newE^.inf.nomergrupp:5,' '); //печать данных об элементах по полям*

*write('|',newE^.inf.fio:15);*

*write('|',newE^.inf.propusk:5,' ');*

*write('|', newE^.inf.opravdan:5,' |');*

*write(newE^.inf.neopravdan:7,' |');*

*write(newE^.inf.summa:6,' |');*

*writeln(newE^.inf.procent:5:2,' |');*

*write('| | | | | | | |');*

*writeln;*

*newE:=newE^.nr; //перевод указателя*

*end;*

*if otpech=0 then*

*otpech:=otpech+n;*

*writeln('|------|---------------|----------|---------|-----------|--------|-------|');*

*writeln('Push the -enter- to the ExiT'); //для выхода ентер*

*end;*

*{------------------------------------------------------------------------------}*

*{---------------DOBAVLENIE ELEMENTA--------------------------------------------}*

*{------------------------------------------------------------------------------}*

*procedure dob;*

*var*

*pop:ukaz;*

*p,i:integer;*

*begin*

*gotoxy(25,25);*

*textcolor(red);*

*writeln('Push the -\*- to the ExiT'); //для выхода*

*textcolor(white);*

*gotoxy(25,27);*

*writeln('Input the name of a new student'); //ввод имени студента*

*gotoxy(25,28);*

*readln(z.fio); //чтение имени*

*{Uslovie vihoda}*

*if z.fio='\*'then//условие выхода*

*exit;*

*for i:=1 to 15-length(z.fio) do//достраиваем имя до 15 знаков*

*z.fio:=z.fio+' ';*

*gotoxy(25,29); //чтение всех данных о студенте с клавиатуры*

*writeln('Input tne nomber of grupp');*

*gotoxy(25,30);*

*readln(z.nomergrupp);*

*gotoxy(25,31);*

*writeln('Input the omount of absence');*

*gotoxy(25,32);*

*readln(z.propusk);*

*gotoxy(25,33);*

*writeln('Input the omount of validly absence');*

*gotoxy(25,34);*

*readln(p);*

*while p>z.propusk do//если число пропусков меньше числа пропусков по уваж причине*

*begin*

*textcolor(red);*

*writeln('Thi is impossible:');*

*textcolor(black);*

*writeln('The omount of validly absence more then just omount of absence');*

*write('Push the -enter- and input theomount of validly absence again');*

*readln; textcolor(white);*

*gotoxy(25,33);*

*writeln('Input the omount of validly absence');*

*gotoxy(25,34); //очищение экрана*

*writeln(' ');*

*writeln(' ');*

*writeln(' ');*

*write(' ');*

*gotoxy(25,34);*

*readln(p); //по новой вводим и читаем*

*end;*

*z.opravdan:=p;*

*{-------------------dobavlenie-------------------------}*

*//происходит так же как при организации*

*pop:=verh; //указатель на начало*

*while pop<>nil do//пока не конец списка*

*if z.fio>= pop^.inf.fio//сравнение по фамилиям*

*then*

*pop:=pop^.nr//перевод указателя*

*else*

*break; //прервать цикл*

*new(newE); //для нового элемента*

*newE^.inf.fio:=z.fio;*

*newE^.inf.propusk:=z.propusk;*

*newE^.inf.opravdan:=z.opravdan;*

*newE^.inf.nomergrupp:=z.nomergrupp;*

*newE^.inf.neopravdan:= newE^.inf.propusk- newE^.inf.opravdan;*

*{----------------v konec------------------------------}*

*if pop=nil then*

*begin*

*right^.nr:=newE;*

*newE^.nl:=right;*

*newE^.nr:=nil;*

*right:=newE;*

*kolvo:=kolvo+1;*

*end*

*else*

*{----------------v na4alo------------------------------*

*if pop=verh then*

*begin*

*pop^.nl:=newE;*

*newE^.nr:=pop;*

*newE^.nl:=nil;*

*kolvo:=kolvo+1;*

*verh:=newE;*

*left:=newE;*

*end*

*else*

*{----------------v seredinu------------------------------}*

*begin*

*pop^.nl^.nr:=newE;*

*newE^.nl:=pop^.nl;*

*pop^.nl:=newE;*

*newE^.nr:=pop;*

*kolvo:=kolvo+1;*

*end;*

*end;*

*{---------------------------------------------------------------------------------}*

*{---------------------------------------UDALENIE----------------------------------}*

*{---------------------------------------------------------------------------------}*

*procedure udal;*

*var*

*i:integer;*

*begin*

*gotoxy(20,25);*

*textcolor(black);*

*writeln('Input the name of a student which could be delete ');*

*textcolor(white);*

*gotoxy(25,27);*

*readln(z.fio); //ввод и чтение имени*

*for i:=1 to (15-length(z.fio)) do//достаиваем фамилию до 15 знаков*

*z.fio:=z.fio+' ';*

*newE:=verh;*

*while newE<>right^.nr do//пока не конец списка*

*begin*

*{------------------------Uslovie esli posledniy---------------------}*

*if newE^.inf.fio=z.fio then*

*begin*

*if newE=right then*

*begin*

*temp:=newE;*

*newE:=newE^.nl;*

*newE^.nr:=nil;*

*right:=newE;*

*dispose(temp);*

*kolvo:=kolvo-1;*

*clrscr;*

*textcolor(green);*

*gotoxy(24,25);*

*writeln('The student has been deleted'); //удалили*

*readln;*

*exit;*

*end*

*else*

*{-----------------Uslovie esli 1-iy el-t udalyayut--------------}*

*if newE =left then*

*begin*

*temp:=newE;*

*newE:=newE^.nr;*

*newE^.nl:=nil;*

*verh:=newE;*

*dispose(temp);*

*kolvo:=kolvo-1;*

*clrscr;*

*textcolor(green);*

*gotoxy(24,25);*

*writeln('The student has been deleted'); //удалили*

*readln;*

*exit;*

*end*

*{-------------------Esli lyuboy no ne 1-iy i ne posledniy-------------}*

*else*

*begin*

*temp:=newE;*

*newE^.nl^.nr:=temp^.nr;*

*newE^.nr^.nl:=newE^.nl;*

*dispose(temp);*

*kolvo:=kolvo-1;*

*clrscr;*

*textcolor(green);*

*gotoxy(24,25);*

*writeln('The student has been deleted'); //удалили*

*readln;*

*exit;*

*end;*

*end;*

*newE:=newE^.nr; //перевод указателя*

*end;*

*gotoxy(15,25);*

*clrscr;*

*textcolor(red);*

*gotoxy(15,25);*

*writeln('This student has not been finded !!!'); //не удалили*

*gotoxy(15,27);*

*writeln('Input his name without mistakes again'); //введите имя снова*

*textcolor(white);*

*readln;*

*exit;*

*end;*

*{------------------------------------------------------------------------------}*

*{-------------------------------P O I S K --------------------------------}*

*{------------------------------------------------------------------------------}*

*{------------------------------------------------------------------------------}*

*//смотреть prosmotr\_2 –всё аналогично*

*procedure poisk(x:ukaz;imya:string);*

*var*

*n,i:integer;*

*c:char;*

*begin*

*clrscr;*

*n:=0;*

*i:=0;*

*newE:=verh;*

*while (newE<>right^.nr) do begin//пока не конец списка*

*if newE^.inf.fio=imya then*

*inc(i); // подсчёт людей с таким именем*

*newE:=newE^.nr;*

*end;*

*if i=0 then begin clrscr;gotoxy(27,25); textcolor(red);writeln('This student does not EXIST'); exit; end; //такого нет*

*newE:=x;*

*textcolor(white);*

*textbackground(blue);*

*writeln('|------|---------------|----------|---------|-----------|--------|-------|');*

*writeln('|#grupp| Name | amount of| validly | unvalidly | summa |percent|');*

*writeln('| | | | absence | | | |');*

*writeln('|------|---------------|----------|---------|-----------|--------|-------|');*

*while (newE<>right^.nr) and (n<=6) do*

*begin*

*inc(n);*

*if newE^.inf.fio=z.fio then*

*textbackground(red) //выделение элемента красным если это тот*

*else*

*textbackground(blue);*

*write('|', newE^.inf.nomergrupp:5,' '); //печать информации по полям*

*write('|',newE^.inf.fio:15);*

*write('|',newE^.inf.propusk:5,' ');*

*write('|', newE^.inf.opravdan:5,' |');*

*write(newE^.inf.neopravdan:7,' |');*

*write(newE^.inf.summa:6,' |');*

*writeln(newE^.inf.procent:5:2,' |');*

*write('| | | | | | | |');*

*textbackground(blue);*

*writeln;*

*newE:=newE^.nr;*

*end;*

*if otpech=0 then//изменение установок для печати*

*otpech:=otpech+n;*

*textcolor(white);*

*writeln('|------|---------------|----------|---------|-----------|--------|-------|');*

*writeln('Push the -enter- to the ExiT'); //для выхода ентер*

*end;*

*{-------------------------------------------------------------------------}*

*procedure obrabotka;*

*var*

*pop:ukaz;*

*x:integer; //промежуточная величина*

*begin*

*pop:=verh; //указатель на начало*

*x:=0;*

*while pop<>nil do//пока не конец списка*

*begin*

*pop^.inf.neopravdan:= pop^.inf.propusk- pop^.inf.opravdan; //подсчёт неоправданных пропусков*

*if pop^.inf.propusk=0 then pop^.inf.procent:=0//если пропуски=0 то и процент =0*

*else*

*pop^.inf.procent:=( pop^.inf.neopravdan/ pop^.inf.propusk)\*100; //подсчёт процента*

*x:=x+pop^.inf.neopravdan; //нахождение промежуточной величины*

*pop^.inf.summa:=x;*

*pop:=pop^.nr;*

*end;*

*gotoxy(25,25);*

*writeln('Processing data is over push the -enter-'); //завершилась обработка*

*readln;*

*end;*

*{------------------------------------------------------------------------}*

*{------------------prosmotr----------------------------------------------}*

*{------------------------------------------------------------------------}*

*//смотреть prosmotr\_2 – всё аналогично ,только добавляются нолвые поля.*

*procedure prosmotr(x:ukaz);*

*var*

*n:integer;*

*c:char;*

*begin*

*clrscr;*

*n:=0;*

*newE:=x;*

*writeln('|------|---------------|----------|---------|');*

*writeln('|#grupp| Name | amount of| validly |');*

*writeln('| | | absence | |');*

*writeln('|------|---------------|----------|---------|');*

*while (newE<>right^.nr) and (n<=6) do*

*begin*

*inc(n);*

*write('|', newE^.inf.nomergrupp:5,' ');*

*write('|',newE^.inf.fio:15);*

*write('|',newE^.inf.propusk:5,' ');*

*writeln('|', newE^.inf.opravdan:5,' |');*

*write('| | | | |');*

*writeln;*

*newE:=newE^.nr;*

*end;*

*if otpech=0 then*

*otpech:=otpech+n;*

*writeln('|------|---------------|----------|---------|');*

*writeln('Push the -enter- to the Exit');*

*end;*

*procedure klavishi(nom:integer);*

*var*

*c:char;*

*left:ukaz;*

*begin*

*otpech:=0; //обнуляем*

*left:=verh; //указатель на начало*

*case nom of//выбор*

*1:prosmotr(left);*

*2:prosmotr\_2(left);*

*3:begin clrscr;gotoxy(20,25);textcolor(white);Writeln('Input the name of a student which cold been finded');*

*textcolor(white);gotoxy(25,27);readln(z.fio);for i:=1 to (15-length(z.fio)) do*

*z.fio:=z.fio+' ';left:=verh;poisk(left,z.fio); end; //установки перед процедурой поиска*

*end;*

*while true do//бесконечный цикл*

*begin*

*c:=readkey; //чтение клавиши*

*if ord(c)=13 then//если ентер-выход*

*case nom of*

*1:exit;*

*2:exit;*

*3:exit;*

*end*

*else// иначе*

*if c=#0 then*

*begin*

*c:=readkey;*

*if c=#80 then//если вниз*

*begin*

*if left=right then case nom of//если конец списка*

*1:prosmotr(left); //выбор*

*2:prosmotr\_2(left);*

*3:poisk(left,z.fio);*

*end*

*else*

*begin*

*if (kolvo-otpech=0) then//если всё отпечатали*

*case nom of*

*1:prosmotr(left); //выбор*

*2:prosmotr\_2(left);*

*3:poisk(left,z.fio);*

*end*

*else*

*begin*

*otpech:=otpech+1; //увеличение числа отпечатанных на 1*

*left:=left^.nr; //перевод указателя*

*case nom of//выбор*

*1:prosmotr(left);*

*2:prosmotr\_2(left);*

*3:poisk(left,z.fio);*

*end*

*end;*

*end;*

*end;*

*if c=#72 then//если вверх*

*begin*

*if left^.nl=nil//если список из 1-го эл-та*

*then case nom of//выбор*

*1:prosmotr(left);*

*2:prosmotr\_2(left);*

*3:poisk(left,z.fio);*

*end*

*else//иначе*

*begin*

*otpech:=otpech-1; //уменьшение числа отпечатанных на1*

*left:=left^.nl; //перевод указателя*

*case nom of//выбор*

*1:prosmotr(left);*

*2:prosmotr\_2(left);*

*3:poisk(left,z.fio);*

*end*

*end;*

*end*

*else//иначе не вверз не вниз не ентер*

*case nom of//выбор*

*1:prosmotr(left);*

*2:prosmotr\_2(left);*

*3:poisk(left,z.fio);*

*end*

*end*

*end;*

*end;*

*{-------------------------------------------------------------------------}*

*{---------------------------ZAPIS V FAIL----------------------------------}*

*{--------------------------------------------------------------------------}*

*procedure zapis\_v\_fail;*

*var*

*t:text;*

*pop:ukaz;*

*d:string;*

*begin*

*gotoxy(25,25);*

*writeln('Input the name of output file'); //ввод имени файла*

*gotoxy(25,26);*

*readln(d);*

*assign(t,'d'); //установка связи сним*

*rewrite(t); //открытие в режиме записи*

*newE:=verh; //указатель на начало*

*//запись в файл шапки*

*writeln(t,'|-------|---------------|----------|---------|----------|--------|-------|');*

*writeln(t,'|#grupp | Name | amount of| validly | unvalidly| summa |percent|');*

*writeln(t,'| | | | absence | | | |');*

*writeln(t,'|-------|---------------|----------|---------|----------|--------|-------|');*

*while (newE<>nil) do//пока не конец сиска*

*begin//запись таблицы*

*write(t,'|', newE^.inf.nomergrupp:5,' ');*

*write(t,'|',newE^.inf.fio:15);*

*write(t,'|',newE^.inf.propusk:5,' ');*

*write(t,'|', newE^.inf.opravdan:5,' |');*

*write(t,newE^.inf.neopravdan:7,' |');*

*write(t,newE^.inf.summa:6,' |');*

*writeln(t,newE^.inf.procent:5:2,' |');*

*writeln;*

*newE:=newE^.nr; //перевод указателя*

*end;*

*//запись в файл последней строки*

*write(t,'|-------|---------------|----------|---------|----------|--------|-------|');*

*clrscr;*

*gotoxy(25,25);*

*writeln('File has been written '); //всё записали*

*readln;*

*close(t); //закрытие файла*

*end;*

*{--------------------------------------MENYU--------------------------------}*

*procedure okno(t:integer);{t-nomer stroki}*

*var*

*p:integer;*

*begin*

*clrscr;*

*textcolor(white); //смена цвета*

*textbackground(blue);*

*writeln('МНННННННННННННННННННННННННННННННННННННН№');*

*gotoxy(25,25);*

*if t=1 then//печать пунктов меню и если t равно номеру их строки то выделение чёрным цветом(для каждого)*

*textbackground(black);*

*writeln('є 1.Input the filename. є');*

*textbackground(blue);*

*gotoxy(25,26);*

*if t=2 then*

*textbackground(black);*

*writeln('є 2.Processing data. є');*

*textbackground(blue);*

*gotoxy(25,27);*

*if t=3 then*

*textbackground(black);*

*writeln('є 3.Delete the student. є');*

*textbackground(blue);*

*gotoxy(25,28);*

*if t=4 then*

*textbackground(black);*

*writeln('є 4.Add student. є');*

*textbackground(blue);*

*gotoxy(25,29);*

*if t=5 then*

*textbackground(black);*

*writeln('є 5.Watch є');*

*textbackground(blue);*

*gotoxy(25,30);*

*if t=6 then*

*textbackground(black);*

*writeln('є 6.Search the student. є');*

*textbackground(blue);*

*gotoxy(25,31);*

*if t=7 then*

*textbackground(black);*

*writeln('є 7.Write output file. є');*

*textbackground(blue);*

*gotoxy(25,32);*

*if t=8 then*

*textbackground(black);*

*writeln('є 8.ExiT. є');*

*textbackground(blue);*

*gotoxy(25,33);*

*writeln('ИННННННННННННННННННННННННННННННННННННННј');*

*textbackground(blue);*

*end;*

*{------------------------------------------------------}*

*{-------------------------------------------------------------------------}*

*{----------------------------------------------------------------------------------}*

*{------------------------------OSNOVNAYA-------------------------------------------}*

*{----------------------------------------------------------------------------------}*

*begin*

*x:=1; //номер строки*

*textbackground(blue);*

*okno(x);*

*while true do//бесконечный цикл*

*begin*

*c:=readkey; //чтение клавиши*

*if ord(c)=13 then//ентер*

*case x of//выбор процедур по их номеру пунктов*

*1:begin clrscr;org;if sozdan=true then klavishi(1);end;*

*2:begin clrscr;obrabotka;end;*

*3:begin clrscr;udal;end;*

*4:begin clrscr;dob;end;*

*5:begin clrscr;obrabotka;klavishi(2);end;*

*6:begin clrscr;klavishi(3);end;*

*7:begin clrscr;zapis\_v\_fail;end;*

*8:begin clrscr;halt;end;*

*end;*

*if c=#0*

*then*

*begin*

*c:=readkey;*

*if c=#80 then//если вниз*

*begin*

*x:=x+1; //увеличение номера строки на 1*

*if x>8 then//если он больше 8*

*begin*

*x:=1; //то он будет =1*

*okno(x); //печать меню*

*end*

*else*

*okno(x); //печать меню*

*end*

*else*

*if c=#72 then//если вверх*

*begin*

*x:=x-1; //уменьшение номера строки на 1*

*if x<1 then//меньше1*

*begin*

*x:=8; //присвоить значение 8*

*okno(x); //печать меню*

*end*

*else*

*okno(x); //печать меню*

*end*

*else*

*okno(x); //печать меню*

*end*

*else*

*okno(x); //печать меню*

*end;*

*end.*