|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Сибирский Государственный Унивеверситет Телекамуникаций и Информатики | | | | | |
|  |  | | | Кафедра ВТ и ТС | |
| **КУРСОВАЯ РАБОТА** | | | | | |
|  | | ДИСЦИПЛИНА: *Информатика*  ПРЕПОДАВАТЕЛЬ: *Моренкова Ольга Илинична*  ТЕМА: *Модули, записи и файлы.*  ИСПОЛНИТЕЛЬ: *Грязина Дмитрия Владимировича* | | |
|  | | |  | | |
| **Новосибирск**  **2000г.** | | | | | |

С О Д Е Р Ж А Н И Е

Введение 3

Глава 1. Модули в Турбо Паскале. 6

1.1. Назначение и структура модуля 6

1.2. Компиляция модулей. 9

Глава 2. Алгоритмы 11

2.1. Вызывающая программа 12

2.2.Процедура формирование массива А 13

2.3. Процедура вывода массива А 14

2.4. Процедура произведение элементов массива А 15

2.5. Функция поиска максимального элемента векторного массива С…….…….16

Глава 3. Модуль основной программы 17

Глава 4. Основная программа 19

Глава 5. Результаты первой части курсовой работы 21

Глава 6. Записи. 22

# Глава 7. Файлы.………………………………………………………………………..24

7.1. Операция с файлами как единое целое. 24

7.2. Структура файлов 26

Глава 8. Задание второй части курсовой работы 30

Глава 9. Алгоритм основной программы 31

Глава 10. Основная программа 32

Глава 11. Результаты второй части курсовой работы……………………………….34

Заключение 35

Список использованных источников: 36

Введение

Целью курсового проектирования является приобретение и закрепление навыков в организации вычислительных процессов и программирования на алгоритмическом языке ПАСКАЛЬ. Основной акцент в работе уделяется рассмотрению модульного программирования, использованию процедур и функций. Кроме того, большое внимание уделяется работе с файлами, подробно рассматриваются функции открытия и закрытия файла. Полученные знания помогут в будущем более эффективно подходить к процессу разработки программного обеспечения.

Язык программирования Pascal был разработан в 1968—1971 гг. Никлаусом Виртом в Цюрихском Институте информатики (Швейца­рия). Первоначальная цель разработки языка диктовалась необходи­мостью инструмента "для обучения программированию как система­тической дисциплине". Однако очень скоро обнаружилась чрезвычай­ная эффективность языка Pascal в самых разнообразных приложениях, от решения небольших задач численного характера до разработки сложных программных систем — компиляторов, баз данных, операци­онных систем и т.п. К настоящему времени Pascal принадлежит к группе наиболее распространенных и популярных в мире языков программирования. Существуют многочисленные реализации языка практически для всех машинных архитектур; разработаны десятки диалектов и проблемно-ориентированных расширений языка Pascal;

обучение программированию и научно-технические публикации в значительной степени базируются на этом языке.

Характеристика и особенности языка. Существует ряд объективных причин, обусловивших выдающийся успех языка Pascal. Среди них в первую очередь необходимо указать следующие:

1. Язык в естественной и элегантной форме отразил важнейшие современные концепции технологии разработки программ: развитая система типов, ориентация на принципы структурного программиро­вания, поддержка процесса пошаговой разработки.

2. Благодаря своей компактности, концептуальной целостности и ортогональности понятий, а также удачному первоначальному описа­нию, предложенному автором языка, Pascal оказался весьма легок для изучения и освоения. В противоположность громоздким многотомным описаниям таких языков, как PL/I, Cobol, FORTRAN, достаточно полное описание языка Pascal занимает около 30 страниц текста, а его синтаксические правила можно разместить на одной странице.

3. Несмотря на относительную простоту языка, он оказался пригоден для весьма широкого спектра приложений, в том числе для разработки очень больших и сложных программ, например операционных систем.

4. Pascal весьма технологичен для реализации практически для всех, в том числе и нетрадиционных, машинных архитектур. Утверждается, что разработка Pascal-транслятора "почти не превышает по трудоем­кости хорошую дипломную работу выпускника вуза". Благодаря этому для многих ЭВМ существует несколько различных реализации языка, отражающих те или иные практические потребности программистов.

Язык Pascal стандартизован во многих странах. В 1983 году был принят международный стандарт (ISO 7185:1983). Готовится (сведения на 1989 год) стандарт СЭВ на Pascal.

Рассмотрим основные особенности языка Pascal.

1. Pascal является традиционным алгоритмическим языком програм­мирования, продолжающим линию Algol-60. Это означает, что про­грамма на языке Pascal представляет собой специально организованную последовательность шагов по преобразованию данных, приводящую к решению некоторой задачи. Это отличает Pascal от так называемых непроцедурных языков типа Prolog, по существу представляющих собой формализмы для записи начальных условий некоторой задачи и синтезирующих решение посредством встроенных механизмов логиче­ского вывода.

2. Язык Pascal содержит удобные средства для представления данных. Развитая система типов позволяет адекватно описывать данные, под­лежащие обработке, и конструировать структуры данных произвольной сложности. Pascal является типизированным языком, что означает фиксацию типов переменных при их описании, а также строгий контроль преобразований типов и контроль доступа к данным в соответствии с их типом (как на этапе компиляции, так и при исполнении программ).

3. Набор операторов языка Pascal отражает принципы структурного программирования и позволяет записывать достаточно сложные алго­ритмы в компактной и элегантной форме.

Pascal является процедурным языком с традиционной блочной структурой и статически определенными областями действия имен. Процедурный механизм сочетает в себе простоту реализации и исполь­зования и гибкие средства параметризации.

4. Синтаксис языка достаточно несложен. Программы записываются в свободном формате, что позволяет сделать их наглядными и удобными для изучения.

# Глава 1. Модули в Турбо Паскале.

## 1.1. Назначение и структура модуля

Модуль - это автономно компилируемая программная единица, включающая в себя различные компоненты раздела описаний (типы, константы, переменные, процедуры и функции) и, возможно, некоторые исполняемые операторы инициирующей части. В модулях Turbo PASCAL явным образом выделяется некоторая "видимая" интерфейсная часть, в которой сконцентрированы описания глобальных типов, констант и переменных, а также приводятся заголовки глобальных процедур и функций. Появление объектов в интерфейсной части делает их доступными для других модулей и основной программы. Тела процедур и функций располагаются в исполняемой части модуля, которая может быть скрыта от пользователя. Модули представляют собой прекрасный инструмент для разработки библиотек прикладных программ и мощное средство модельного программирования. Важная особенность модуля заключается в том, что компилятор Turbo PASCAL размещает их программный код в отдельном сегменте. Модуль имеет следующую структуру:

UNIT <имя>;

INTERFACE

<интерфейсная часть>

IMPLEMENTATION

<исполняемая часть>

BEGIN

<инициирующая часть>

END.

Здесь UNIT - зарезервированное слово, начинает заголовок

модуля;

<имя> - имя модуля (должно совпадать с именем файла, в

который помещен модуль);

IMPLEMENTATION - зарезервированное слово (выполнение), начинает исполняемую часть; BEGIN - зарезервированное слово, начинает инициирующую часть (часть модуля BEGIN <инициирующая часть> необязательна); END - зарезервированное слово, признак конца модуля.

Заголовок модуля состоит из зарезервированного слова UNIT и следующего за ним имени модуля. Для правильной работы Turbo PASCAL и возможности подключения средств, облегчающих разработку крупных программ, это имя должно совпадать с именем файла, в который помещается исходный текст модуля. Имя модуля служит для его связи с другими модулями и основной программой. Эта связь устанавливается специальным приложением USES <список модулей>. Здесь USES - зарезервированное слово; <список модулей> - список модулей, с которыми устанавливается связь, элементами списка являются имена модулей, отделяемые друг от друга запятыми. Если объявление USES используется, то оно должно открывать раздел описаний основной программы. Модули могут использовать другие модули. Предложение USES в модулях может следовать либо сразу за зарезервированным словом INTERFACE, либо сразу за словом IMPLEMENTATION. Интерфейсная часть открывается зарезервированным словом INTERFACE. В этой части содержатся объявления всех глобальных объектов модуля (типов, констант, переменных и подпрограмм), которые должны стать доступными основной программе и (или) другим модулям. При объявлении глобальных подпрограмм в интерфейсной части указывается только их заголовок. Объявление подпрограмм в интерфейсной части автоматически сопровождается их компиляцией с использованием дальней модели памяти. Таким образом, обеспечивается доступ к подпрограммам из основной программы и других модулей. Порядок появления различных разделов объявлений и их количество может быть произвольным. Исполняемая часть начинается зарезервированным словом IMPLEMEN- TATION и содержит описания подпрограмм, объявленных в интерфейсной части. В ней могут объявляться локальные для модуля объекты - вспомогательные типы, константы, переменные и блоки, а также метки, если они используются в инициирующей части. Описанию подпрограммы, объявленной в интерфейсной части модуля, в исполняемой части должен предшествовать заголовок, в котором можно опускать список формальных переменных (и тип результата для функции), так как они уже описаны в интерфейсной части. Но если заголовок программы приводится в полном виде, т.е., со списком формальных параметров, то он должен совпадать с заголовком, объявленным в интерфейсной части. Инициирующая часть завершает модуль. Она может отсутствовать вместе с начинающим ее словом BEGIN или быть пустой - тогда за BEGIN сразу следует признак конца модуля (слово END и следующая за ним точка). В инициирующей части размещаются исполняемые операторы, содержащие некоторый фрагмент программы. Эти операторы исполняются до передачи управления основной программе и обычно используются для подготовки ее работы. Например, в них могут инициироваться переменные, открываться нужные файлы, могут устанавливаться связи с другими ПК по коммуникационным каналам и т.п. Не рекомендуется делать инициирующую часть пустой, лучше ее опустить: пустая часть содержит пустой оператор, которому будет передано управление при запуске программы.

## 1.2. Компиляция модулей.

В среде Turbo PASCAL имеются средства, управляющие способом компиляции модулей и облегчающие разработку крупных программных проектов. В частности, определены три режима компиляции: COMPILE, MAKE и BUILD. Режимы отличаются только способами связи компилируемого модуля и основной программы с другими модулями, объявленными в приложении USES. При компиляции модуля или основной программы в режиме COMPILE компилируется только файл, находящийся в активном окне. По умолчанию предполагается, что используемые модули уже откомпилированы и результаты помещены в одноименные файлы с расширением .TPU; файл с расширением .TPU (от английского Turbo Pаscal Unit) создается в результате компиляции модуля. В режиме МАКЕ компилятор проверяет наличие TPU-файлов для каждого объявленного модуля. Если какой-либо из файлов не обнаружен, то система пытается отыскать одноименный файл с расширением .PAS, т.е. файл с исходным текстом модуля. Если PAS-файл найден, то происходит его компиляция. Кроме того, в этом режиме система следит за возможными изменениями исходного текста любого используемого модуля. Если в PAS-файл (исходный текст модуля) внесены какие-либо изменения, то независимо от того, есть ли уже в каталоге соответствующий TPU-файл или нет, система осуществляет его компиляцию перед компиляцией основной программы. Если изменения внесены в интерфейсную часть модуля, то будут перекомпилированы также и все другие модули, обращающиеся к нему. Режим МАКЕ существенно облегчает процесс разработки крупных программ с множеством модулей. В режиме BUILD существующие TPU-файлы игнорируются, и система пытается отыскать и компилировать соответствующий PAS-файл для каждого объявленного в предложении USES модуля. После компиляции в режиме BUILD программист может быть уверен в том, что учтены все сделанные им изменения в любом из модулей.

Подключение модуля к основной программе и их возможная компиляция осуществляется в порядке их объявления в предложении USES. При переходе к очередному модулю система предварительно отыскивает все модули, на которые он ссылается. Ссылки модулей друг на друга могут образовывать древовидную структуру любой сложности, однако запрещается явное или косвенное обращение модуля к самому себе. Дело в том, что Turbo PASCAL разрешает ссылки на частично откомпилированные модули, что приблизительно соответствует опережающему описанию подпрограммы. Если интерфейсные части любых двух модулей независимы, Turbo PASCAL сможет идентифицировать все глобальные идентификаторы в каждом из модулей, после чего откомпилирует тела модулей обычным способом.

# Глава 2. Задание первой части курсовой работы

##### Разработать модуль (схемы алгоритмов и программы), содержащие процедуры и функции, решающие поставленную задачу :

##### Вариант № 21

##### Вычислить произведение элементов каждого столбца матрицы :

##### Определить максимальное значение произведения :

##### Разработать вызывающую программу (алгоритм и текст программы), предусмотрев процедуру очистки экрана и вывода информации на экран с текстовыми комментариями. Например :

##### 

##### Курсовую работу выполнил студент группы А – 61 Иванов Роман Борисович

##### Вариант № 21

##### Разработанный модуль содержит следующие программы :

##### Процедуру формирования элементов матрицы в интервале –1.0 до 1.0.

##### Функцию определения суммы элементов той строки, где находиться максимальный элемент.

##### Процедуру вывода элементов матрицы.

##### Результаты :

##### Сформированная матрица :

##### ------------------------------

##### ------------------------------

Сумма элементов строки, где находится максимальный элемент =…….

## 2.1. Вызывающая программа

Данные о студенте и о программе

Начало

VIVOD (A)

Вывод сформи

рованой матрицы и результатов

MAX (C)

PR (A)

FORM (A)

Конец

## 2.2.Процедура формирования массива А

# PROCEDURE FORM

###### Конец

a[i,j]:=random\*2-1;

## J =1,n

# I= 1,n

randomize

###### Вход

## 2.3.Процедура ввывода массива А

# PROCEDURE VIVOD

###### Конец

a[i,j]:6:2

## J =1,n

# I= 1,n

###### Вход

## 2.4.Процедура произведения элементов массива А

PROCEDURE PR

###### C[i] = 1

###### Конец

C[i] = a[i,j]\*c[i]

###### J = 1,n

###### I =1,n

Вход

## 2.5.Функция поиска максимального элемента в векторном массиве С

FUNCTION MAX

###### Конец

###### M =C[1]

MAX =M

###### C[i] > m

###### I = 1,n

###### M = -2

###### Вход

# Глава 3. Модуль основной программы

unit kurp1;

interface

const n=5;

type matr=array[1..n,1..n] of real;

vect=array[1..n] of real;

procedure form(var a:matr);

procedure vivod (var a:matr);

function max(c:vect):real;

procedure pr(a:matr;var c:vect);

implementation

procedure vivod (var a:matr);

var i,j:integer;

begin

for i:=1 to n do

begin

for j:=1 to n do

write (a[i,j]:6:2);

writeln;

end;

writeln;

end;

procedure form(var a:matr);

var i,j:integer;

begin

randomize;

for i:=1 to n do

for j:=1 to n do

a[i,j]:=random\*2-1;

end;

function max( c:vect):real;

var i:integer;

m:real;

begin

m:=-2;

for i:=1 to n do

if c[i]>m then m:=c[i];

max:=m

end;

procedure pr(a:matr;var c:vect);

var i,j:integer;

begin

c[i]:=1;

for i:=1 to n do

begin

for j:=1 to n do

c[i]:=a[j,i]\*c[i];

writeln('Произведение ',i,'-го столбца =',c[i]:6:4);

end;

writeln;

end;

end.

# Глава 4. Основная программа

program kursovik;

uses crt,kurp1;

var a:matr;

c:vect;

begin

clrscr;

window (45,2,75,7) ;

writeln ('Курсовая работа по информатике') ;

writeln('Факультет АЭС группа А-92');

writeln ('студента: Грязина Дмитрия ');

window(1,1,80,25);

writeln('вариант 21');

window(10,8,80,20);

writeln('Разработанный модуль содержит следующие программы:');

writeln('1.Процедуру формирования элементов матрицы в интервале от 1.0 до 1.0.');

writeln('2.Процедуру вывода матрицы на экран.');

writeln('3.Процедуру определения произведения каждого столбца матрицы.');

writeln('4.Функцию определения максимального произвелдения.');

window(1,1,80,25);

gotoxy(10,24);

writeln('Для продолжения нажмите пробел');

clrscr;

form(a);

writeln('Сформированная матрица');

vivod(a);

pr(a,c);

writeln('Максимальное произведение =',max(c):6:4);

readln;

end.

# Глава 5. Результаты первой части курсовой работы

Сформированная матрица

-0.68 0.22 0.58 -0.62 -0.14

0.25 -0.11 -0.90 0.41 -0.57

0.24 -0.69 -0.84 0.28 0.71

-0.46 -0.26 0.16 0.49 -0.73

0.60 0.17 0.28 -0.77 -0.61

Произведение 1-го столбца =0.0112

Произведение 2-го столбца =-0.0007

Произведение 3-го столбца =0.0187

Произведение 4-го столбца =0.0264

Произведение 5-го столбца =0.0243

Максимальное произведение =0.0264

# Глава 6. Записи.

Запись – это структура данных, состоящая из фиксированного числа компонентов, называемых полями записи. В отличие от массива, компоненты (поля) записи могут быть различного типа. Чтобы можно было ссылаться на тот или иной компонент записи, поля именуются.

Структура обьявления типа записи такова:

<имя типа> = RECORD <пс. полей> END

Здесь <имя типа> - правельный индификатор;

RECORD, END – зарезервированные слова (запись, конец);

<сп. полей> - список полей; представляют собой последовательность

разделов записи, между которыми ставится точка с запитой.

Каждый раздел записи состоит из одного или нескольких индификаторов полей, отделяемых друг от друга запятыми. За индификатором ставится двоеточие и описание типа поля например:

type

birthday = record

day, month **:** byte;

year **:** word:

end;

var

a,b : Birthday;

**. . . . . . . .**

В этом примере тип Birthday есть запись с полями DAY, MONTH, и YEAR; переменные А и В содержат записи типа Birthday.

Как в массиве, значения переменных типа записи можно присваивать другим переменным того же типа, например:

a : = b;

К каждому из компонентов записи можно получить доступ, если использовать составное имя, т.е. указать имя переменной, затем точку и имя поля:

a.day := 27; b.year := 1991;

Чтобы упростить доступ к полям записи, используется оператор присоединения WITH:

WITH <переменная> DO <оператор>

Здесь WITH, DO – ключевые слова (с, делать);

<переменная> - имя переменной типа запись, за которым,

возможно, следует список вложенных полей;

<переменная> - любой оператор Турбо Паскаля.

Например:

with c.bd do month := 9;

Имена полей должны быть уникальными в пределах той записи, где они обьявлены, однако, если записи содержат поля – записи, т.е. вложена одна в другую, имена могут повторяться на разных уровнях вложенности.

.

# Глава 7. Файлы.

Под файлом понимается именованная область памяти на магнитном носителе, используемая для долговременного хранения некоторой логиче­ски связанной информации. В эту область внешней памяти можно как поместить определенные данные, так и извлечь их из нее. Эти действия имеют общее название - ввод-вывод.

## 7.1. Операция с файлами как единое целое.

Программирование доступа к файлу в языке Turbo Pascal начинается с описания файловой переменной (переменной файлового типа). Это опи­сание строится с помощью служебного слова file, после которого (после служебного слова of) может быть указан тип отдельных элементов файла, например:

var

F : file of integer;

При необходимости можно отдельно определить некоторый файловый тип, которым в дальнейшем пользоваться при описании файловых пере­менных:

type

RecordFile = file of record

А, В : real;

end;

var

Fl, F2 : RecordFile;

Очень важно различать собственно файл (область памяти на магнит­ном носителе с некоторой информацией) и переменную файлового типа в некоторой Turbo Pascal- программе. Считается, что файловая переменная являетсяпредставителем некоторого дискового файла в программе. Для того, чтобы реализовать доступ к файлу па магнитном диске, программа должнасвязать его с файловой переменной. Для этого необходимо сопос­тавить переменную с именем файла. Это имя представляется в виде стро­ки, содержащей полное имя файла и, быть может, цепочку объемлющих каталогов и обозначение дисковода. Такая строка носит название маршру­та (или пути) к файлу и формируется по общим правилам MS-DOS.

Связывание маршрута и файловой переменной производится стан­дартной процедурой Assign, которой передаются два параметра: имя пе­ременной файлового типа и строка с именем (маршрутом) дискового файла, например:

Assign (F,'с: \books\book4\test3 .раз ' ) ;

Помимо маршрута, во втором параметре может быть указано условное (символическое) обозначение некоторого «псевдофайла» MS-DOS, то есть файла, связанного с конкретным физическим устройством. Допускаются имена, в основном совпадающие с принятыми в MS-DOS обозначениями (‘con’ – консоль, то есть экран дисплея и/или клавиатура, ‘lpti’ или ‘prn’ – печатающее устройство и т. д.).

До тех пор, пока файловая переменная не связана с каким-либо дис­ковым файлом, никакие операции с ней в программе невозможны, Заме­тим, что можно связать файловую переменную сеще не существующимдисковым файлом. Это может быть оправдано для случая последующего создания Turbo Pascal-программой файла с данным именем с помощью некоторой системной процедуры.

После того, как файловая переменная с помощью процедуры Assign связана с конкретным дисковым файлом, можно выполнить любую до­пустимую операцию с ним. Здесь рассмотрим две операции общего ха­рактера, позволяющие переименовывать и уничтожать существующие дисковые файлы. Такие операции выполняются системными подпрограм­мами Rename и Erase со следующими спецификациями:

Rename ( var f; nevname : string );

Erase ( var f );

Основным параметром этих процедур является файловая переменная (она может идентифицировать файл любого типа, и к ней должна была быть ранее применена процедура Assign), а процедуре Rename дополни­тельно передается новое имя файла. Действие указанных процедур осно­вывается на соответствующих DOS-функциях и функционально эквива­лентно командам ren и del командного процессора.

## 7.2. Структура файлов

*7.2.1Открытие и закрытие файлов*

В этом и следующем разделах рассматриваются файлы традиционной для языка Pascal структуры, которые описываются в программе в терми­нах составляющих компонент. Кроме таких файловых переменных (они также называются "типизированными" файлами), в языке Turbo Pascal имеются также файлы специального типа - текстовые файлы, которые введены для поддержки одного практически важного частного случая файловых структур, и нетипизированные файлы, которые используются для более эффективной организации обменов на низком уровне.

Важнейшим моментом при использовании переменных файловых ти­пов является интерпретация файла как потенциально бесконечного спи­ска значений одного и того же (базового) типа. Например, описание пе­ременной F вида

var

F : file of integer;

понимается как определение в программе под именем F списка неопреде­ленного количества целых чисел, расположенного на некотором внешнем запоминающем устройстве (например, на магнитном диске).

С каждой переменной файлового типа также связано понятие теку­щего указателя файла. Текущий указатель можно понимать как скрытую переменную (то есть переменную, неявно описанную вместе с файловой

переменной), которая обозначает ("указывает" на) некоторый конкретный элемент файла. Таким образом, можно графически так представить ин­терпретацию файла, связанного с переменной F: 0 1 2 N

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Целое | Целое | Целое | . . . . . | Целое | . . . |

Текущий указатель —————

Как правило, все действия с файлом (чтение из файла, запись в файл) производятся поэлементно, причем в этих действиях участвует тот эле­мент файла, который обозначается текущим указателем. В результате со­вершения операций текущий указатель может перемещаться, настраиваясь на тот или иной элемент файла. Все элементы файла считаются пронуме­рованными; начальный элемент имеетнулевой номер. Имеется возмож­ность явной установки текущего указателя на некоторый элемент файла.

Заметим, что один и тот же внешний файл в различных Turbo Pascal-программах (или даже в различных частях одной и той же программы) может интерпретироваться по-разному, например, как последовательность целых чисел, или как последовательность символов, или как последова­тельность некоторых записей, и т. д. Эта возможность очень важна и час­то используется в работе с файлами сложной структуры. Например, в процессе работы с внутренним представлением файлов баз данных (в частности, с DBF-файлами) может понадобиться интерпретировать файл по-разному в зависимости от того, какую его компоненту (заголовок или содержательную часть) мы обрабатываем.

Нужно также отметить, что все операции с файлами в языке Turbo Pascal реализованы с использованием базовых средств операционной сис­темы; в частности, понятие текущего указателя файла поддержано в MS-DOS. Однако все перемещения по файлу (как явное управление указателем, так и сдвиги при обменах) выполняются в терминах байтов. Таким образом, все аспекты, связанные с типизацией содержимого фай­лов, представляют собой системную надстройку языка Turbo Pascal.

Для выполнения некоторой операции по вводу-выводу необходимо прежде всегооткрыть дисковый файл (напомним, что предварительно нужно связать файловую переменную с дисковым файлом с помощью процедуры Assign). Под открытием в данном случае понимается подготов­ка к обмену с файлом со стороны программы, заключающаяся прежде всего в поиске файла на внешнем носителе, установке его текущего указа­теля на определенный (как правило, нулевой элемент) и формировании буфера в оперативной памяти, посредством которого будут производиться обмены.

Обычные типизированные файлы открываются с помощью одной из двух системных процедур - Reset или Rewrite, единственным параметром

которых в данном случае является файловая переменная. Обе эти проце­дуры открывают файл для обмена. Разница между ними заключается в начальных действиях с файлом.

Процедура Reset предполагает, что открываемый дисковый файл **уже** существует, в противном случае возникает ошибка. Процедура Rewrite допускает, что открываемый файл можетеще не существовать; в этом случае она создаст заданный файл. Если же файл существует, то Rewrite очищает его. Повторим, что в обоих случаях текущий указатель файла устанавливается на его нулевой элемент. Следует иметь в виду, что обе операции допускают в дальнейшем как чтение из файла,так и запись в него (это будет обсуждаться в следующем разделе).

Разумеется, две введенные процедуры носят в основном подготови­тельный характер, однако можно указать практически важные случаи их вполне самостоятельного использования. Так, процедура Rewrite может применяться для стирания содержимого файла без его удаления с диска или для создания нового файла.

Процедура close завершает действия с файлом, который указывается в качестве ее единственного параметра. При этом буфер, образованный при открытии файла, ликвидируется. После этого файловую переменную можно связать посредством процедуры Assign с каким-либо другим дис­ковым файлом. Заметим, что при окончании работы всей программы происходит автоматическое закрытие всех файлов, открытых в программе. Однако хорошим правилом является явное закрытие файлов процедурой close после окончания работы с ними.

Помимо закрывающей процедуры close, в системе Turbo Pascal также имеется дополнительная процедура Flush (ее параметр - файловая пере­менная). Flush используется для завершения обменов с файлом без его закрытия. Обмены с файлами всегда реализуются через некоторый буфер в оперативной памяти, поэтому в процессе записи на файл последние записываемые элементы могут еще находиться ("застрять") в буфере. Процедура Flush вызывает принудительный сброс этих элементов в файл. Эта процедура имеет смысл в том случае, если с файлом проводились операции записи, и используется редко, так как процедура close произ­водит такие же действия.

# Глава 8. Задание по второй части курсовой работы

# В файле содержаться справочные сведенья о предприятиях-задолжниках по уплате налогов

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| № | ИНН | Наименование предприятия | Средство на начало года | Начислено налогов в отчет. период | Уплачено налогов в отч. Период |
| ……... | ………….. | ……………… | ……………. | ……………… | ………….. |
| ……... | ………….. | ……………… | ……………. | ……………… | ………….. |
| Сведенья на конец отчетного периода | | | | | |
| ………………….. | | ………………. | …………… | ……………… | …………. |
| ………………….. | | ………………. | ……………. | …………….. | …………. |

# 

# Тип полей:

№ - тип INTEGER[3]

ИНН – тип STRING[12]

Наименование

Предприятия – тип STRING[20]

Остальные поля – INTEGER

Разработать алгоритм и программу для обработки файла согласно варианту задания

Вариант №26

Выдать информацию о предприятиях, которые не начисляют налоги в отчетном периоде.

Сведенья выдать в виде таблицы с соответствующим заголовком.

НАПРИМЕР :

СВЕДЕНЬЯ О ЗАДОЛЖНИКАХ > 20 тыс.

# Глава 9. Алгоритм второй части курсовой работы

начало

i:= 0

max:=0

# Вывод упорядоченных сведений

Данные о студенте и о программе

f,’table.txt’

’

конец

Открыть f

Заголовок

not eof(f)

f,x

Not Eof(f)

Max:=sng

N1:=i

Read(f,x)

WITH x1[i]

Sng>0

# I:=i+1

B[i]:=x

Sng>max

# Глава 10. Основная программа

# program part2;

# uses crt;

type table = record

n : integer ;

inn : string[12];

Name : string[20];

Sng : integer;

Nop : integer;

Uop : integer;

Skop : integer;

End;

Var

f : file of table;

x : table;

begin

clrscr;

window(45,2,75,7);

textcolor (5) ;

writeln (‘Курсовая работа по информатике ’) ;

writeln (‘Факультет АЭС группа А-92 ’) ;

writeln (‘студента: Грязина Д.’) ;

window (1, 1, 80, 25) ;

gotoxy (30, 5) ;

writeln (‘Часть 2’) ;

window(10, 8, 80, 20) ;

writeln (‘В файле table.txt содержатся справочные сведения’) ;

writeln (‘о предприятиях-задолжниках по уплате налогов ’) ;

writeln ;

writeln (‘Задание’) ;

writeln (‘Выдавать полные сведения о предприятиях, которые не ’) ;

writeln (‘начисляли налоги в отчетный период.’) ;

window(1, 1, 80, 25);

gotoxy(10, 24);

textcolor (4 + blink);

writeln (‘Для продолжения нажмите Enter ’) ;

readln ;

textcolor (10) ;

clrscr ;

assign (f, table.txt) ;

reset (f) ;

writeln (‘Cведения о предприятиях, которые не начисляли ’) ;

writeln (‘налоги в отчетный период ’) ;

writeln (‘+----------------------------------+’) ;

writeln (‘| № | ИНН | Наименование | 1 | 2 | 3 | 4 |’) ;

writeln (‘ | ---+----------+----------------+----+----+----+-----| ’) ;

while not Eof(f) do

begin

read (f, x) ;

with x do

if Nop = 0 then

begin

writeln (‘| ‘№ : 3’ | ‘,Inn : 12,’ | ’,Name : 20,’ | ’ ,Sng :4,’ | ’,Nop : 4,’ | ’,Uop : 4,’|’ ) ;

writeln (Skop : 4, ‘ | ’) ;

end;

end;

writeln (‘+-----------------------------------------------------------------------+’) ;

writeln ;

writeln (‘1- Сведения на начало года ’) ;

writeln (‘2 – Начислено налогов в отчетный период ’) ;

writeln (‘3 – Уплачено налогов в отчетный прериод ’) ;

writeln (‘4 – Сведения на конец отчетного преиода ’) ;

writeln ;

readln ;

end.**Глава 11. Результаты второй части курсовой работы**

Сведения о предприятиях которые не начисляли

налоги в отчетный период.

+-----------------------------------------------------------------------+

¦ № ¦ ИНН ¦ Наименование ¦ 1 ¦ 2 ¦ 3 ¦ 4 ¦

¦-----+-----------------+------------------------+----+----+----+-----¦

¦ 3 ¦ 54090903618 ¦ Детский сад №275¦ -8 ¦ 0 ¦ 8 ¦ 0 ¦

¦ 10 ¦ 54091032618 ¦ Школа №25 ¦ -7 ¦ 0 ¦ 7 ¦ 0 ¦

+------------------------------------------------------------------------+

1 - Сведения на начало года

2 - Начислено налогов в отчетный период

3 - Уплачено налогов в отчетный период

4 - Сведения на конец отчетного периода

# Заключение

Полученный мною практический опыт применительно к программированию на ПАСКАЛЕ позволяет использовать эффективное модульное программирование. Кроме того, изучив представленный в курсовом проектировании материал, я приобрел практические навыки использования процедур, функций. Наряду с этим я научился правильно использовать файловые функции операционной системы MS-DOS, что является одним из основных вопросов программирования.

### **Список использованных источников:**

Фаронов В. В., Турбо Паскаль 7.0., Учебное пособие. – М.: «Нолидж», 1999.

Бляшкин И. И. Буров А. А. Новые возможности Турбо Паскаля 6.0. – СПб.: Издательство «Макет»,1992.

Фаронов Н.Н. Програмирование на персональных ЭВМ в среде Турбопаскаль. – М.: Издательство МГТУ, 1990.