Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет информатики и радиоэлектроники»

Филиал

«Минский радиотехнический колледж»

Учебная дисциплина «Структуры и алгоритмы обработки данных»

**Инструкция**

по выполнению лабораторной работы

«Реализация алгоритмов внешней сортировки»

Минск

2018

**Лабораторная работа № 3**

**Тема работы: «Реализация алгоритмов внешней сортировки»**

**1. Цель работы**

Изучение принципов работы с файлами как структурой данных.

**2. Задание**

1. Дан файл, в котором через пробел записаны натуральные числа. Вывести на экран суммы цифр каждого числа.

2. Создать файл X, компонентами которого являются элемента массива M. Переписать из файла X в файл Y все компоненты, которые стоят на четных местах. Прочитать файл Y.

M = (0,1; 5,6; 0.8; 2,4; 7,3; 6,8; 0,5; 2,3;)

3. Создать типизированый файл (в соответствии с вариантом), и заполнить информацией, не менее 5 записей указанного типа. Файл сохранить с именем Input.txt. Применить сортировку файла по любому полю. Результат поместить в файл с именем Output.txt.

**Вариант 1.** В файле разместить информацию о книгах в библиотеке: фамилия автора, название книги, издательство, год издания, цена.

**Вариант 2.** В файле разместить информацию о футбольных командах, участвующих в чемпионате мира: страна, название команды, фамилия главного тренера, рейтинг команды.

**3. Оснащение работы**

ПК, Pascal.

**4. Основные теоретические сведения**

**Внешняя сортировка** – это сортировка данных, которые расположены на внешних устройствах и не вмещающихся в оперативную память.

**Файлом** называется совокупность данных, записанная во внешней памяти под определенным именем.

Файл является динамической структурой данных, размер которой может меняться в процессе выполнения над ним каких-либо действий (размер файла может быть равен нулю, что соответствует пустому файлу).

Для хранения информации используют различные запоминающиеся устройства: гибкие магнитные диски, оптические диски, винчестера и другие носители. Использование внешних запоминающих устройств хранения информации является наиболее надежным и удобным способом хранения информации.

С одной стороны, **файл** – это именованная область внешней памяти, содержащая какую-либо информацию. Файл в таком понимании называют физическим файлом, то есть существующим физически на некотором материальном носителе информации. Организация хранения информации зависит от конкретного носителя, и подробно рассматриваться нами не будет.

*Для организации работы с физическими файлами в языках программирования предусмотрены специальные структуры данных – тип данных файл.*

С другой стороны, **файл** – это одна из многих структур данных используемых в программировании. Файл в таком понимании называют логическим файлом, т.к. в этом случае он представляет собой лишь логическую модель хранения информации не зависящую от организации конкретного физического файла, *то есть существующим только в нашем логическом представлении при написании программы.* В программах логические файлы представляются файловыми переменными определенного типа.

*Файл может находиться в различных состояниях:*

1. *файл закрыт;*
2. *файл открыт для чтения;*
3. *файл открыт для записи.*

Файлы становятся доступны программе только после выполнения особой процедуры открытия файла. Эта процедура заключается в связывании ранее объявленной файловой переменной с именем существующего или вновь создаваемого файла, а также в указании направления обмена информацией: чтение из файла или запись в него.

*В любой момент времени программе доступен только один элемент файла, на который ссылается текущий указатель. Часто позицию размещения доступного элемента называют текущей позицией.*

Для доступа к файлу описывается специальная файловая переменная, которая считается представителем файлов в Паскаль-программе (чаще всего ее обозначают как F). Если файл состоит из записей, дополнительно описывается переменная для доступа к полям записи (обозначим ее R)

*Синтаксис:*

**type**

**<имя типа> = <тип компонентов>;**

**var**

**<F>: file of <имя типа>;**

**<R>: <имя типа>;**

*Переменные файлового типа могут быть описаны в программе либо явно в разделе переменных* ***var****, либо с использованием раздела типов* ***type****.*

*Объявление файлов в разделе переменных имеет вид:*

**текстовый файл**

**var A: text;**

**типизированный файл**

**var <имя файла>: file of <базовый тип элементов>;**

**файл без типа**

**var A: file;**

**var**

**f1: file of char; {типизированный файл}**

**f2: file of integer; {типизированный файл}**

**f3: file; {нетипизированный файл}**

**f: text; {текстовый файл}**

Существует несколько видов файлов. Файлы в Turbo Pascal классифицируются по двум признакам:

- по типу (логической структуре): типизированные, текстовые, нетипизированные;

- по методу доступа к элементам: последовательно доступа, прямого доступа.

**Текстовые** файлы трактуются как последовательности строк переменной длины. В конце каждой строки ставится специальный признак конца строки EOLN.

**Типизированные** файлы – последовательности компонент определенного типа. Длина любого элемента типизированного файла строго постоянна, что дает возможность организовать прямой доступ к каждому компоненту.

*Чтобы можно было работать с типизированным файлом, необходимо, как и для текстовых файлов, сначала связать имя файловой переменной с внешним именем файла (оператор* ***Assign****). Затем нужно открыть его (используются операторы* ***Reset*** *и* ***Rewrite****, но не* ***Append****). Операторы* ***Reset*** *и* ***Rewrite*** *открывают файл и для чтения, и для записи (а не только для чтения или только для записи, как при использовании текстовых файлов). Отличие их в том, что оператор* ***Reset*** *открывает только существующий файл (если такого файла нет, будет сгенерирована ошибка времени выполнения). С другой стороны, оператор* ***Rewrite*** *создает новый файл (если файл с таким именем уже имеется, то он будет уничтожен и создан заново). При открытии файла с ним связывается текущий указатель файла, который позиционируется на его первый элемент. Оперировать можно только тем элементом файла, на который ссылается указатель файла. При чтении или записи элемента файла происходит автоматическое перемещение указателя на следующий элемент. Чтение из типизированного файла производится оператором* ***Read*** *(но не ReadLn), а запись в него — оператором* ***Write*** *(но не WriteLn). Однако следует помнить, что в списке вывода оператора* ***Write*** *могут быть только переменные. Типы элементов файла и типы переменных в списках ввода-вывода должны быть согласуемы по присваиванию. Элементами типизированных файлов могут быть числовые, символьные, булевы, строковые значения, массивы, записи, но не файлы или структуры с файловыми элементами.*

Узнать количество элементов типизированного файла (размер файла) можно с помощью функции **FileSize**, для которой используется следующий синтаксис:

**FileSize(имя\_файла)**

*Например, если переменная* ***k*** *имеет тип* ***LongInt****, а* ***f*** *– файловая переменная типизированного файла, то оператор* ***k := FileSize(f)****, записывает в переменную* ***k*** *размер файла* ***f.***

Элементы типизированного файла нумеруются с нуля (порядковый номер последнего элемента файла на единицу меньше размера файла). Чтобы узнать, на каком элементе располагается указатель файла, используют функцию **FilePos**:

**FilePos(имя\_файла)**

Текущим положением указателя можно управлять, для чего служит процедура **Seek,** которая использует следующий синтаксис:

**Seek(имя\_файла, номер\_элемента)**

Второй параметр (тип **LongInt**) задает номер элемента (отсчет от 0), на который должен переместиться указатель файла. Рассмотрим несколько примеров.

*Перейти к пятому (фактически шестому) элементу файла* ***f****:*

**Seek(f, 5);**

*Перейти к предыдущему элементу:*

**Seek(f, FilePos(f)-1);**

*Перейти в конец файла:*

**Seek(f, FileSize(f)-1);**

Как и для текстовых файлов, можно использовать функцию **Eof**(имя\_файла), которая возвращает значение **True**, если текущий указатель расположен на признаке конца файла (т. е. при выполнения равенства **FilePos(имя\_файла) = FileSize(имя\_файла)**).

*Процедура* ***Seek*** *и функции* ***FilePos*** *и* ***FileSize*** *позволяют легко осуществлять коррекцию элементов типизированного файла, имя которого указано в качестве её параметра, начиная с элемента, на котором расположен указатель. Однако уничтожить элемент внутри файла нельзя, для этого файл должен быть перезаписан.*

Типизированные файлы создаются в результате работы какой-либо программы.

*Пример записи данных в типизированный файл:*

**type**

**t\_subscriber = record**

**surname: string[20];**

**tel: LongInt;**

**end;**

**var**

**subscriber: t\_subscriber;**

**f: file of t\_subscriber;**

**i: Integer;**

**begin**

**Assign(f,'notebook.dat');**

**Rewrite(f);**

**for i:=1 to 5 do begin**

**with subscriber do begin**

**Write('Surname: ');**

**ReadLn(surname);**

**Write('Phone: ');**

**ReadLn(tel);**

**end;**

**Write(f, subscriber);**

**end;**

**Close(f);**

**end.**

*Пример последовательного доступа к типизированному файлу:*

**type**

**t\_subscriber = record**

**surname: string[20];**

**tel: LongInt;**

**end;**

**var**

**subscriber: t\_subscriber;**

**f: file of t\_subscriber;**

**s: string[7];**

**begin**

**Assign(f,'notebook.dat');**

**Reset(f);**

**while not Eof(f) do begin**

**Read(f, subscriber);**

**with subscriber do begin**

**str(tel,s);**

**if Copy(s,1,2) = '33' then**

**tel := tel+4000000;**

**end;**

**Seek(f,FilePos(f)-1); // возврат указателя назад**

**Write(f,subscriber);**

**end;**

**Close(f);**

**end.**

*В приведенной программе типизированный файл обрабатывается и как файл последовательного доступа, и как файл произвольного доступа.*

**Нетипизированные** файлы отличаются тем, что для них не указан тип компонент. Такой подход делает нетипизированные файловые переменные совместимыми с файлами любых типов, а также позволяет организовать высокоскоростной обмен данными между оперативной и внешней памятью.

**Файлом последовательного доступа** называется файл, к элементам которого обеспечивается доступ в такой же последовательности, в какой они записывались.

**Файлом прямого (произвольного) доступа** называется файл, доступ к элементам которого осуществляется по адресу элемента.

*Например, для поиска нужного элемента в последовательном файле необходимо, начиная с нулевого, перемещать указатель обработки до тех пор, пока он не будет указывать на искомый элемент, а при поиске нужного элемента в файле прямого доступа достаточно указать номер его позиции. При организации данных в файл последовательного доступа нельзя одновременно читать данные из файла и записывать данные в файл, так как для чтения некоторого элемента этот указатель обработки помещен на данный элемент, а для записи нового элемента этот указатель одновременно должен быть в конце файла.*

Алгоритм, рекомендуемый при работе с файлами:

1. установить соответствие между файлом и файловой переменной;
2. открыть файл с определенной целью (чтение, запись);
3. работа с файлом;
4. закрытие файла.

|  |  |
| --- | --- |
| **Процедуры и функции для всех типов файлов** | |
| **ASSIGN(F,NAME);** | Связь файловой переменной F с внешним файлом NAME.  Каждому файлу в языке ставится в соответствие файловая переменная определенного типа, поэтому перед началом работы с файлом необходимо установить данное соответствие. Данная процедура всегда предшествует другим процедурам работы с файлами, так как впоследствии все другие файловые процедуры будут обращаться не к имени файла на диске, а к данной файловой переменной, связанной с реальным файлом. Данную процедуру нельзя использовать для уже открытого файла. Для этого необходимо с помощью процедуры CLOSE закрыть этот файл. |
| **CLOSE(F);** | Закрытие открытого файла.  Операция закрытия является логическим окончанием работы с любым открытым файлом. Использование данной процедуры позволяет устранить связь файловой переменной с внешним файлом, установленную с помощью процедуры ASSIGN. |
| **ERASE(F);** | Процедура, удаляющая неоткрытый внешний файл любого типа, задаваемый переменной F.  Удаление осуществляется только для реально существующего файла. |
| **RENAME(F,NEWNAME);** | Переименование внешнего файла.  Осуществляется только для уже реально существующего файла. |
| **RESET(F[, SIZE]);** | Открывает уже существующий файл, с которым связна файловая переменная F, и указатель текущего компонента файла настраивается на начало файла. Необязательный параметр целого типа SIZE используется только с файлами без типа и задает размер пересылаемого элемента информации в байтах (по умолчанию 128) |
| **REWRITE(F[, SIZE]);** | Открывает новый пустой файл, и ему присваивает имя, заданное процедурой ASSIGN.  Если файл с таким именем уже существует, то он уничтожается |
| **EOF(F);** | Конец файла |
| **Процедуры и функции для работы с текстовыми файлами** | |
| **APPEND(F);** | Открытие файла для добавления в конец информации |
| FLUSH(F); | Отображает буфер выходного файла.  Информация из буфера вывода записывается в файл, с которым связана файловая переменная F. |
| **READ(F, <список>);** | Чтение из файла |
| **READLN(F, <список>);** | Чтение строки из файла |
| SETTEXTBUF(F, BUF); | Назначение буфера ввода-вывода.  Для файла, с которым связана файловая переменная F, назначается буфер ввода-вывода в виде переменной BUF любого типа. |
| **WRITE(F, <список>);** | Запись в файл |
| **WRITELN(F, <список>);** | Запись строки в файл |
| EOLN(F); | Конец строки файла |
| SEEKEOF(F); | Конец файла |
| SEEKEOLN(f); | Конец строки файла.  Пропускает в конце строки символы пробела и табуляции. |

*Для связи файла в коде программы и действительного файла на внешнем носителе используется процедура* ***ASSIGN:***

**assign(myfile,'c:\text.txt');**

*где* ***myfile****— имя переменной (объявленной ранее в области* ***var****), ассоциированной с файлом;*

***c:\text.txt*** *— путь к реальному файлу.*

*Первый аргумент процедуры assign в паскаль — переменная, второй – путь к файлу на диске.*

*Пример:*

*В файле*input.txt *записаны числа (каждое — с новой строки), их количество не превышает* 100*. Необходимо* [*отсортировать*](http://labs.org.ru/pascal-5/#i-11) *их по возрастанию и записать в файл* output.txt*.*

*Трудности:*

- для сортировки необходим массив, для того чтобы одновременно работать со всеми числами;

- неизвестно общее количество чисел.

*Алгоритм решения:*

*- объявляем массив для 100 элементов;*

*- открываем файл на чтение, просчитываем количество чисел, заполняя массив, сохраняем количество в N;*

*- сортируем N элементов массива;*

*- записываем результат в файл.*

*Фрагмент решения:*

**{ Определяем глобальные переменные: }**

**var A: array[1..100] of integer;**

**f: text;**

**N, i: integer;**

**{ Определяем функцию, считывающую числа из файла, и**

**записывающую их в массив. Функция возвращает кол-во элементов массива: }**

**function ReadFromFile: integer;**

**var i: integer;**

**begin**

**assign(f, 'input.txt');**

**...;{ открытие файла в режиме чтения }**

**i := 0;**

**while (...) and (...) do begin**

**i := i + 1;**

**readln(...,...);**

**close(f);**

**ReadFromFile := i;**

**end;**

**{ Основная программа }**

**Begin**

**N := ReadFromFile ;**

**{ сортировка N элементов по возрастанию }**

**{ ... }**

**{ запись отсортированного массива в файл: }**

**assign(..., ...);**

**...;{ открытие файла в режиме записи }**

**for i:=1 to N do**

**writeln(..., ...);**

**close(f);**

**end.**

*Основой большинства алгоритмов внешней сортировки является* ***принцип******слияния*** *двух упорядоченных последовательностей в единый упорядоченный набор.*

К наиболее известным алгоритмам внешних сортировок относятся:

- **сортировки слиянием** (простое слияние и естественное слияние);

- **улучшенные сортировки** (многофазная сортировка и каскадная сортировка).

*Основным понятием при использовании* *внешней сортировки является понятие серии*. **Серия** (упорядоченный отрезок) – это последовательность элементов, которая упорядочена по ключу.

Количество элементов в серии называется **длиной серии**. Серия, состоящая из одного элемента, упорядочена всегда. Последняя серия может иметь длину меньшую, чем остальные серии файлов. Максимальное количество серий в файле **N** (все элементы не упорядочены). Минимальное количество серий одна (все элементы упорядочены).

*В основе большинства методов внешних сортировок лежит процедура слияния и процедура распределения*. **Слияние** – это процесс объединения двух (или более) упорядоченных серий в одну упорядоченную последовательность при помощи циклического выбора элементов, доступных в данный момент. **Распределение** – это процесс разделения упорядоченных серий на два и несколько вспомогательных файла.

**Фаза** – это действия по однократной обработке всей последовательности элементов. **Двухфазная сортировка** – это сортировка, в которой отдельно реализуется две фазы: распределение и слияние. **Однофазная сортировка** – это сортировка, в которой объединены фазы распределения и слияния в одну.

**Двухпутевым** слиянием называется сортировка, в которой данные распределяются на два вспомогательных файла. **Многопутевым** слиянием называется сортировка, в которой данные распределяются на N (N > 2) вспомогательных файлов.

**Естественное слияние** – это сортировка, при которой всегда сливаются две самые длинные из возможных серий.

**Несбалансированное слияние** – это естественное слияние, у которого после фазы распределения количество серий во вспомогательных файлах отличается друг от друга более чем на единицу.

**Простое слияние** – это одна из сортировок на основе слияния, в которой длина серий фиксируется на каждом шаге.

**Сбалансированное слияние** – это естественное слияние, у которого после фазы распределения количество серий во вспомогательных файлах отличается друг от друга не более чем на единицу.

**Общий алгоритм сортировки слиянием**

Сначала серии распределяются на два или более вспомогательных файлов. Данное распределение идет поочередно: первая серия записывается в первый вспомогательный файл, вторая – во второй и так далее до последнего вспомогательного файла. Затем опять запись серии начинается в первый вспомогательный файл. После распределения всех серий, они объединяются в более длинные упорядоченные отрезки, то есть из каждого вспомогательного файла берется по одной серии, которые сливаются. Если в каком-то файле серия заканчивается, то переход к следующей серии не осуществляется. В зависимости от вида сортировки сформированная более длинная упорядоченная серия записывается либо в исходный файл, либо в один из вспомогательных файлов. После того как все серии из всех вспомогательных файлов объединены в новые серии, потом опять начинается их распределение. И так до тех пор, пока все данные не будут отсортированы.

*Выделим* основные характеристики сортировки слиянием:

- количество фаз в реализации сортировки;

- количество вспомогательных файлов, на которые распределяются серии.

*Рассмотрим основные и наиболее важные алгоритмы внешних сортировок более подробно.*

**Сортировка простым слиянием**

Одна из сортировок на основе слияния называется **простым слиянием**.

Алгоритм сортировки простым слияния является простейшим алгоритмом внешней сортировки, основанный на процедуре слияния серий.

В данном алгоритме длина серий фиксируется на каждом шаге. В исходном файле все серии имеют длину 1, после первого шага она равна 2, после второго – 4, после третьего – 8, после k-го шага – 2k.

Алгоритм сортировки простым слиянием:

**Шаг 1**. Исходный файл **f** разбивается на два вспомогательных файла **f1** и **f2**.

**Шаг 2**. Вспомогательные файлы **f1** и **f2** сливаются в файл **f**, при этом одиночные элементы образуют упорядоченные пары.

**Шаг 3.** Полученный файл **f** вновь обрабатывается, как указано в шагах 1 и 2. При этом упорядоченные пары переходят в упорядоченные четверки.

**Шаг 4.** Повторяя шаги, сливаем четверки в восьмерки и т.д., каждый раз удваивая длину слитых последовательностей до тех пор, пока не будет упорядочен целиком весь файл.

После выполнения **i** проходов получаем два файла, состоящих из серий длины **2i.** Окончание процесса происходит при выполнении условия **2i>=n.** Следовательно, процесс сортировки простым слиянием требует порядка O(log n) проходов по данным.

Признаками конца сортировки простым слиянием являются следующие условия:

- длина серии не меньше количества элементов в файле (определяется после фазы слияния);

- количество серий равно 1 (определяется на фазе слияния).

- при однофазной сортировке второй по счету вспомогательный файл после распределения серий остался пустым.

*Заметим, что для выполнения****внешней сортировки методом простого слияния*** *в оперативной памяти требуется расположить всего лишь две переменные – для размещения очередных элементов (записей) из вспомогательных файлов. Исходный и вспомогательные файлы будут* ***O(log n)*** *раз прочитаны и столько же раз записаны.*

**Сортировка естественным слиянием**

Сортировка, при которой всегда сливаются две самые длинные из возможных последовательностей, является естественным слиянием. В данной сортировке объединяются серии максимальной длины.

Алгоритм сортировки естественным слиянием:

**Шаг 1.** Исходный файл **f** разбивается на два вспомогательных файла **f1** и **f2**. Распределение происходит следующим образом: поочередно считываются записи ai исходной последовательности (неупорядоченной) таким образом, что если значения ключей соседних записей удовлетворяют условию **f(ai)<=f(ai+1)**, то они записываются в первый вспомогательный файл **f1**. Как только встречаются **f(ai)>f(ai+1)**, то записи **ai+1** копируются во второй вспомогательный файл **f2**. Процедура повторяется до тех пор, пока все записи исходной последовательности не будут распределены по файлам.

**Шаг 2.** Вспомогательные файлы **f1** и **f2** сливаются в файл **f**, при этом серии образуют упорядоченные последовательности.

**Шаг 3.** Полученный файл **f** вновь обрабатывается, как указано в шагах 1 и 2.

**Шаг 4.** Повторяя шаги, сливаем упорядоченные серии до тех пор, пока не будет упорядочен целиком весь файл.

Символ **"`"** обозначает признак конца серии.

Признаками конца сортировки естественным слиянием являются следующие условия:

- количество серий равно 1 (определяется на фазе слияния);

- при однофазной сортировке второй по счету вспомогательный файл после распределения серий остался пустым.

*Естественное слияние, у которого после фазы распределения количество серий во вспомогательных файлах отличается друг от друга не более чем на единицу, называется сбалансированным слиянием, в противном случае – несбалансированное слияние.*

[Procedure](http://pascal.net.ru/procedure) **MergeSort(name:** [string](http://pascal.net.ru/string)**;** [var](http://pascal.net.ru/var) **f:** [**text**](http://pascal.net.ru/file)**);**

[Var](http://pascal.net.ru/var) **s1,s2,a1,a2,where,tmp:** [**integer**](http://pascal.net.ru/Integer)**;**

**f1,f2:** [**text**](http://pascal.net.ru/file)**;**

[Begin](http://pascal.net.ru/begin)

**s1:=5; s2:=5; {Можно задать любые числа, которые запустят цикл while}**

[**Assign**](http://pascal.net.ru/Assign)**(f,name);**

[**Assign**](http://pascal.net.ru/Assign)**(f1,'{имя 1-го вспомогательного файла}');**

[**Assign**](http://pascal.net.ru/Assign)**(f2,'{имя 2-го вспомогательного файла}');**

[While](http://pascal.net.ru/while) **(s1>1)** [and](http://pascal.net.ru/DIV) **(s2>=1)** [do](http://pascal.net.ru/do)

[begin](http://pascal.net.ru/begin)

**where:=1;**

**s1:=0; s2:=0;**

[**Reset**](http://pascal.net.ru/Reset)**(f);** [**Rewrite**](http://pascal.net.ru/Rewrite)**(f1);** [**Rewrite**](http://pascal.net.ru/Rewrite)**(f2);**

[**Read**](http://pascal.net.ru/Read)**(f,a1);**

[**Write**](http://pascal.net.ru/Write)**(f1,a1,' ');**

[While](http://pascal.net.ru/while)[not](http://pascal.net.ru/DIV)[**EOF**](http://pascal.net.ru/Eof)**(f)** [do](http://pascal.net.ru/do)

[begin](http://pascal.net.ru/begin)

[**read**](http://pascal.net.ru/Read)**(f,a2);**

[If](http://pascal.net.ru/if) **(a2<a1)** [then](http://pascal.net.ru/if)

[begin](http://pascal.net.ru/begin)

[Case](http://pascal.net.ru/case) **where** [of](http://pascal.net.ru/of)

**1:** [begin](http://pascal.net.ru/begin)

**where:=2;**

[**inc**](http://pascal.net.ru/Inc)**(s1);**

[End](http://pascal.net.ru/end)**;**

**2:** [begin](http://pascal.net.ru/begin)

**where:=1;**

[**inc**](http://pascal.net.ru/Inc)**(s2);**

[End](http://pascal.net.ru/end)**;**

[End](http://pascal.net.ru/end)**;**

[End](http://pascal.net.ru/end)**;**

[Case](http://pascal.net.ru/case) **where** [of](http://pascal.net.ru/of)

**1:** [**write**](http://pascal.net.ru/Write)**(f1,a2,' ');**

**2:** [**write**](http://pascal.net.ru/Write)**(f2,a2,' ');**

[End](http://pascal.net.ru/end)**;**

**a1:=a2;**

[End](http://pascal.net.ru/end)**;**

[If](http://pascal.net.ru/if) **where=2** [then](http://pascal.net.ru/if)

[**inc**](http://pascal.net.ru/Inc)**(s2)**

[else](http://pascal.net.ru/else)

[**inc**](http://pascal.net.ru/Inc)**(s1);**

[**Close**](http://pascal.net.ru/Close)**(f);** [**Close**](http://pascal.net.ru/Close)**(f1);** [**Close**](http://pascal.net.ru/Close)**(f2);**

[**Rewrite**](http://pascal.net.ru/Rewrite)**(f);** [**Reset**](http://pascal.net.ru/Reset)**(f1);** [**Reset**](http://pascal.net.ru/Reset)**(f2);**

[**Read**](http://pascal.net.ru/Read)**(f1,a1);**

[**Read**](http://pascal.net.ru/Read)**(f2,a2);**

[While](http://pascal.net.ru/while) **(**[not](http://pascal.net.ru/DIV)[**EOF**](http://pascal.net.ru/Eof)**(f1))** [and](http://pascal.net.ru/DIV) **(**[not](http://pascal.net.ru/DIV)[**EOF**](http://pascal.net.ru/Eof)**(f2))** [do](http://pascal.net.ru/do)

[begin](http://pascal.net.ru/begin)

[If](http://pascal.net.ru/if) **(a1<=a2)** [then](http://pascal.net.ru/if)

[begin](http://pascal.net.ru/begin)

[**Write**](http://pascal.net.ru/Write)**(f,a1,' ');**

[**Read**](http://pascal.net.ru/Read)**(f1,a1);**

[End](http://pascal.net.ru/end)

[else](http://pascal.net.ru/else)

[begin](http://pascal.net.ru/begin)

[**Write**](http://pascal.net.ru/Write)**(f,a2,' ');**

[**Read**](http://pascal.net.ru/Read)**(f2,a2);**

[End](http://pascal.net.ru/end)**;**

[End](http://pascal.net.ru/end)**;**

[While](http://pascal.net.ru/while)[not](http://pascal.net.ru/DIV)[**EOF**](http://pascal.net.ru/Eof)**(f1)** [do](http://pascal.net.ru/do)

[begin](http://pascal.net.ru/begin)

**tmp:=a1;**

[**Read**](http://pascal.net.ru/Read)**(f1,a1);**

[If](http://pascal.net.ru/if)[not](http://pascal.net.ru/DIV)[**EOF**](http://pascal.net.ru/Eof)**(f1)** [then](http://pascal.net.ru/if)

[**Write**](http://pascal.net.ru/Write)**(f,tmp,' ')**

[else](http://pascal.net.ru/else)

[**Write**](http://pascal.net.ru/Write)**(f,tmp);**

[End](http://pascal.net.ru/end)**;**

[While](http://pascal.net.ru/while)[not](http://pascal.net.ru/DIV)[**EOF**](http://pascal.net.ru/Eof)**(f2)** [do](http://pascal.net.ru/do)

[begin](http://pascal.net.ru/begin)

**tmp:=a2;**

[**Read**](http://pascal.net.ru/Read)**(f2,a2);**

[If](http://pascal.net.ru/if)[not](http://pascal.net.ru/DIV)[**EOF**](http://pascal.net.ru/Eof)**(f2)** [then](http://pascal.net.ru/if)

[**Write**](http://pascal.net.ru/Write)**(f,tmp,' ')**

[else](http://pascal.net.ru/else)

[**Write**](http://pascal.net.ru/Write)**(f,tmp);**

[End](http://pascal.net.ru/end)**;**

[**Close**](http://pascal.net.ru/Close)**(f);** [**Close**](http://pascal.net.ru/Close)**(f1);** [**Close**](http://pascal.net.ru/Close)**(f2);**

[End](http://pascal.net.ru/end)**;**

[**Erase**](http://pascal.net.ru/Erase)**(f1);**

[**Erase**](http://pascal.net.ru/Erase)**(f2);**

[End](http://pascal.net.ru/end)**;**

**ВНЕШНЯЯ СОРТИРОВКА С ИСПОЛЬЗОВАНИЕМ ОДНОФАЗНОГО ЕСТЕСТВЕННОГО СЛИЯНИЯ**

**program naturalmerge;**

**uses CRT;**

**type**

**item = record**

**key:integer;**

**end;**

**filetype = file of item;**

**var**

**a,b,c,d,e:filetype;**

**z:integer;**

**eor:boolean;**

**procedure create;**

**var**

**i: byte;**

**buf: item;**

**begin**

**rewrite©;**

**randomize;**

**for i:=1 to 20 do begin**

**buf.key:=random(100);**

**write(c,buf);**

**end;**

**close©;**

**end;**

**procedure view(var x:filetype);**

**var buf: item;**

**begin**

**reset(x);**

**if filesize(x)<>0 then begin**

**repeat**

**read(x,buf);**

**write(buf.key,' ');**

**until eof(x); writeln;**

**readkey; end;**

**end;**

**procedure copy(var x,y:filetype);**

**var buf,buf1:item;**

**begin**

**read(x,buf);**

**write(y,buf);**

**if eof(x) then eor:=true**

**else begin**

**read(x,buf1);**

**seek(x,filepos(x)-1);**

**eor:=buf1.key<buf.key**

**end;**

**end;**

**procedure copyrun(var x,y:filetype);**

**begin**

**repeat**

**copy(x,y);**

**until eor;**

**end;**

**procedure distribute;**

**begin**

**reset©;rewrite(a);rewrite(b);**

**repeat**

**copyrun(c,a);**

**if not eof© then copyrun(c,b);**

**until eof©;**

**write('A= '); view(a);**

**write('B= '); view(b);**

**close(a);close(b);close©;**

**end;**

**procedure mergerun (var w,y,x:filetype);**

**var bufa,bufb:item;**

**begin**

**repeat**

**read(w,bufa); seek(w,filepos(w)-1);**

**read(y,bufb); seek(y,filepos(y)-1);**

**if bufa.key<bufb.key then begin**

**copy(w,x);**

**if eor then copyrun(y,x);**

**end**

**else begin**

**copy(y,x);**

**if eor then copyrun(w,x);**

**end;**

**until eor**

**end;**

**procedure merge;**

**label m1,m2;**

**begin**

**m1: z:=1; reset(a);reset(b); rewrite(d); rewrite(e) ;**

**while (not eof(a)) and (not eof(b)) do**

**begin**

**if odd(z) then mergerun(a,b,d) else mergerun(a,b,e);**

**z:=z+1;**

**end;**

**while not eof(a) do begin**

**copyrun(a,d); z:=z+1; end;**

**while not eof(b) do begin**

**copyrun(b,e);z:=z+1; end;**

**if filesize(d)<>0 then write('D= '); view(d);**

**if filesize(e)<>0 then write('E= '); view(e) ;**

**writeln;**

**If (filesize(e)=0) or (filesize(d)=0)**

**then begin writeln('OTSORTIROVAN'); exit; end;**

**if (eof(a)) and (eof(b)) then goto m2;**

**m2: Z:=1;**

**reset(d); reset(e); rewrite(a); rewrite(b);**

**while (not eof(d)) and (not eof(e)) do begin**

**if odd(z) then mergerun(d,e,a) else mergerun(d,e,b);**

**z:=z+1; end;**

**while not eof(d) do begin**

**copyrun(d,b); z:=z+1; end;**

**while not eof(e) do begin**

**copyrun(e,a); z:=z+1; end;**

**if filesize(a)<>0 then write('A= '); view(a);**

**if filesize(b)<>0 then write('B= '); view(b) ;**

**If (filesize(a)=0) or (filesize(b)=0)**

**then begin writeln('OTSORTIROVAN'); readkey;exit; end;**

**if (eof(d)) and (eof(e)) then goto m1;**

**close(a);close(b);**

**close(d); close(e);**

**end;**

**begin {main} clrscr;**

**assign(a,'a.txt');**

**assign(b,'b.txt');**

**assign(c,'c.txt');**

**assign(d,'d.txt');**

**assign(e,'e.txt');**

**create;**

**writeln('Ishodniy massiv ');**

**view©;**

**distribute; z:=0;**

**merge;**

**writeln('++++++++++++++++++++++++++++++++++++++++++++++++++++') ;**

**end.**

**Примеры программ, обработки текстовые файлы:**

1. *Создать простой текстовый файл с именем WORK.TXT:*

**Var f :text;**

**Begin**

**Assign(f, ‘WORK.TXT’);**

**Rewrite(f);**

**Write(f, ‘Простой текстовый файл’);**

**Close(f);**

**End.**

1. *Удалить или переименовать уже существующий файл:*

**Var f: text;**

**ch: char;**

**st: string;**

**begin**

**write(‘Введите имя файла: ’);**

**read(st);**

**assign(f, st);**

**write(‘Удалить файл(У), Переименовать(П), Выход(В)’);**

**readln(ch);**

**case ch of**

**‘У’,’у’: erase(f);**

**‘П’,’п’: begin**

**write(‘Введите новое имя файла: ’);**

**readln(st);**

**rename(f, st);**

**end;**

**‘В’,’в’: halt(1); {Прерывание программы}**

**end;**

**end.**

1. *Считать данные из первого файла, и записать их во второй файл, удвоив каждый элемент через пробел:*

**Var f1, f2: text;**

**x: char;**

**st:string;**

**begin**

**write(‘Введите имя файла: ’);**

**readln(st);**

**assign(f1, st);**

**reset(f1);**

**assign(f2,’result.txt’);**

**rewrite(f2);**

**while not eof (f1) do**

**begin**

**read(f1,x);**

**write(f2, x + ‘ ‘ + x );**

**end;**

**close(f1);**

**close(f2);**

**end.**

1. *Подсчитать количество символов и строк в текстовом файле:*

**Var f: text;**

**s: string;**

**n, k: integer;;**

**begin**

**writeln(‘Введите имя файла: ’);**

**readln(st);**

**assign(f, st);**

**reset(f);**

**n:=0;**

**k:=0;**

**while not eof (f) do**

**begin**

**readln(f, s);**

**n:=n+length(s);**

**k:=k+1;**

**end;**

**writeln(‘в файле ’, n, ‘ символов и ’,k, ‘ строк’ );**

**close(f);**

**end.**

**5. Порядок выполнения работы**

1. Создать файл, в котором содержаться натуральные числа. Данные из файла считать в динамический массив, определить суммы цифр каждого числа и вывести на экран.

2. Создать файл Х, создать массив М и записать в файл Х. Создать файл Y и переписать в него все элементы массива, которые стоят на четных местах. Вывести содержимое файла Y на экран.

3. Создать типизированный файл с именем Input.txt. Внести в файл не менее 5 записей в соответствии с вариантом. Выполнить сортировку слиянием по любому полю. Результат поместить в файл с именем Output.txt. Для сортировки необходимо использовать дополнительные файлы.

**6. Форма отчета о работе**

*Лабораторная работа № \_\_\_*

*Номер учебной группы \_\_\_\_\_\_\_\_\_\_\_\_*

*Фамилия, инициалы учащегося \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Дата выполнения работы \_\_\_\_\_\_\_\_\_\_\_\_\_*

*Тема работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Цель работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Оснащение работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*Результат выполнения работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

**7. Контрольные вопросы и задания**

1. Дайте определение внешней сортировке.

2. В чем отличия между внешней и внутренней сортировками?

3. Когда применяется внешняя сортировка?

4. Назовите алгоритмы внешней сортировки.

**8. Рекомендуемая литература**

1.Ахо, А.В.Структуры данных и алгоритмы / А.В. Ахо, Дж.Хопкрофт, Дж. Д.Ульман. – пер. с англ. – М.: Издательский дом "Вильямс", 2007.-400 с.

2. Вирт,Н. Алгоритмы и структуры данных / Н. Вирт. –СПб.:Невский диалект, 2008. – 352с.

3. Вирт, Н. Алгоритмы и структуры данных с примерами на Паскале / Н. Вирт[перевод с английского Д. Б. Подшивалова]; – 2-е изд., испр., – СПб.: Невский Диалект, 2005. – 352с.

4. Гагарина, Л.Г. Алгоритмы и структуры данных / Л.Г. Гагарина, В.Д. Колдаев. – учеб.пособие – М: Финансы и статистика, 2009. – 304с.

5. Котов, В.М. Алгоритмы и структуры данных: учеб.пособие / В.М. Котов, Е.П. Соболевская, А.А. Толстиков – Минск: БГУ, 2011. – 267с.

6. Макконнелл, Дж. Основы современных алгоритмов / Дж. Макконнелл – 2-е дополненное издание – М.:Техносфера, 2006. – 368с.

7. Окулов, С.М. Программирование в алгоритмах / С.М. Окулов. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2007. – 383с.