Лабораторная работа №4
Тема: Бинарные файлы.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

Файл – это именованный объект, хранящий данные (программа или любая другая информация) на каком-либо носителе. Файл, как и массив, – это совокупность данных.

Отличия файла от массива:

1. Файлы в отличие от массивов располагаются не в оперативной памяти, а на жестких дисках или на внешних носителях, хотя файл может располагаться на так называемом электронном диске (в оперативной памяти).
2. Файл не имеет фиксированной длины, т.е. может увеличиваться и уменьшаться.
3. Перед работой с файлом его необходимо открыть, а после работы – закрыть.

Файловая система – это совокупность файлов и управляющей информации на диске для доступа к файлам. Или по-другому – это совокупность программных средств для доступа к файлам. Существует довольно много файловых систем и правила именования файлов в них могут незначительно отличаться.

Имена файлов состоят из *двух частей*, разделенных точкой: имя файла и расширение. Файлы хранятся в *каталогах (директориях)*. Каталоги могут иметь такие же имена, что и файлы. Допускаются вложенные каталоги (подкаталоги).

Различают два вида файлов: *текстовые и бинарные*.

Текстовые файлы могут быть просмотрены и отредактированы с клавиатуры любым текстовым редактором и имеют очень простую структуру: последовательность ASCII-символов. Эта последовательность символов разбивается на строки, каждая из которых заканчивается двумя кодами: 13, 10 (0xD, 0xA). Примеры известных текстовых фалов: *.bat, *.c, *.asm, *.cpp.

Бинарные файлы – это файлы, которые не имеют структуры текстовых файлов. Каждая программа для своих бинарных файлов определяет собственную структуру.

Библиотека языка C/C++ содержит функции для работы как с текстовыми, так и с бинарными файлами.

Функции открытия и закрытия файла. Перед работой с файлом, его необходимо открыть.

Функция открытия файла *fopen():*

```
FILE *fopen(char *filename, char *mode);
```

где *FILE* – структурный тип, который связан с физическим файлом и содержит всю необходимую информацию для работы с ним (указатель на текущую позицию в файле, тип доступа и др.).

`char *filename` - задает физическое местонахождение (*путь*) и *имя* открываемого файла;

`char *mode` - тип доступа к файлу, который может принимать значения, указанные в таблице.

Функция *fopen()* при успешном открытии файла возвращает указатель на структуру типа *FILE*, называемый *указателем на файл*. Эта структура связана с физическим файлом и содержит всю необходимую информацию для работы с ним (указатель на текущую позицию в файле, тип доступа и др.). Возвращаемое функцией значение нужно сохранить и использовать для ссылки на открытый файл. Если произошла ошибка при открытии файла, то возвращается *NULL*.

Таблица**:** тип доступа к файлам.

“r”	Открыть файл для чтения.
“w”	Открыть файл для записи. Если файл существует, то его содержимое теряется.
“a”	Открыть файл для записи в конец файла. Если файл не существует, то он создается.
“r+”	Открыть файл для чтения и записи. Файл должен существовать.
“w+”	Открыть файл для чтения и записи. Если файл существует, то его содержимое теряется.
“a+”	Открыть файл для чтения и записи в конец файла. Если файл не существует, то он создается.

К комбинациям вышеперечисленных литералов могут быть добавлены также “t” либо “b”:

“t”	Открыть файл в текстовом режиме.
“b”	Открыть файл в бинарном режиме.

Возможны следующие режимы доступа: “*w+b*”, “*wb+*”, “*rw+*”, “*w+t*”, “*rt+*” и др. Если режим не указан, то файл открывается в текстовом режиме.

После работы с файлом он должен быть закрыт функцией *fclose()*.

Для этого необходимо в указанную функцию передать указатель на *FILE*, который был получен при открытии функцией *fopen()*. При завершении программы незакрытые файлы автоматически закрываются системой.

Стандартная последовательность операторов, необходимая для открытия и закрытия файла:

```
#include <stdio.h>

FILE *f;
if(!(f = fopen(“readme.txt”, ”r+t”)))
{
  printf(“Невозможно открыть файл \n”);  return;
}
// Работа с файлом
fclose(f);                    // Закрытие файла
```

Функции чтения/записи в бинарный файл.

```
unsigned fread (void *ptr, unsigned size, unsigned n, FILE *stream);
unsigned fwrite(void *ptr, unsigned size, unsigned n, FILE*stream);
```

где: **ptr* – указатель на буфер;

size – размер блока;

n – количество блоков;

stream* – указатель на структуру ***FILE* открытого файла.

Пример 1: Записать в бинарный файл и прочитать из бинарного файла список абитуриентов, информация об абитуриенте представлена структурой.

```
#include "stdafx.h"
#include <stdio.h>

struct abitur
{ char name[32];
  int mark[3];
};

int _tmain(int argc, _TCHAR* argv[])
{ struct abitur inf;
  int a;
  FILE *f;

  if(!(f=fopen("inf.dat","w+")))
  { printf("Ошибка создания файла\n"); return 0; }
  for(;;)
  { printf("Введите ФИО (пустая строка -- конец списка): ");
    fflush(stdin);
    gets(inf.name);
    if(!inf.name[0]) break;
```

```
        printf("\n Введите три оценки, полученные на экзаменах:
");
        scanf("%d%d%d", &inf.mark[0], &inf.mark[1],
&inf.mark[2]);
        fwrite(&inf, 1,sizeof(inf), f);
      }
      fclose(f);

      printf("\nСписок абитуриентов:\n");
      if(!(f=fopen("inf.dat","r")))
      {  printf("Ошибка создания файла\n"); return 0;}
      while(1)
      {   if(sizeof(inf) != fread(&inf, sizeof(inf), 1,f))
          break;   /* Если не удалось прочитать необходимое
                      количество байт, то заканчиваем чтение */
          printf("%s  %d  %d  %d  \n", inf.name, inf.mark[0],
inf.mark[1], inf.mark[2]);
      }
      fclose(f);
      return 0;
    }
```

Позиционирование в файле. Каждый открытый файл имеет, так называемый *указатель на текущую позицию* в файле (это нечто подобное указателю в памяти). Все операции над файлами (чтение и запись) работают с данными с этой позиции. При каждом выполнении функции чтения или записи указатель смещается на количество прочитанных или записанных байт, т.е. устанавливается сразу за прочитанным или записанным блоком данных в файле. В этом случае осуществляется так называемый *последовательный доступ* к данным, который очень удобен, когда нам необходимо последовательно работать с данными в файле. Это демонстрируется во всех вышеприведенных примерах чтения и записи в файл. Но иногда необходимо читать или писать данные в произвольном порядке, что достигается путем установки указателя на некоторую заданную позицию в файле функцией *fseek()*.

```
int fseek(FILE *stream, long offset, int whence);
```

Параметр *offset* задает количество байт, на которое необходимо сместить указатель соответственно параметру *whence*. Приводим значения, которые может принимать параметр *whence*:

SEEK_SET	0	Смещение выполняется от начала файла.
SEEK_CUR	1	Смещение выполняется от текущей позиции указателя.
SEEK_END	2	Смещение выполняется от конца файла.

Величина смещения может быть как положительной, так и отрицательной, но нельзя смещаться за пределы начала файла.

Такой доступ к данным в файле называют *произвольным*.

Пример 2: *Найти по номеру запись из бинарного файла, который был создан в примере 1.*

```
#include "stdafx.h"
#include <stdio.h>

struct abitur
{ char name[32];
  int mark[3];
};
int _tmain(int argc, _TCHAR* argv[])
{ struct abitur inf;
  int n;
  FILE *f;
  if(!(f=fopen("inf.dat","r")))
  { printf("Ошибка создания файла\n");
    return 0;
  }
  while(1)
  { printf("Введите номер записи (0 - выход): ");
    scanf("%d", &n);
    if(!n) break;
    fseek(f,sizeof(struct abitur)*(n-1),SEEK_SET);
    if(sizeof(inf) != fread(&inf,1,sizeof(inf),f))
      printf("Конец списка\n");
    else
      printf("%s %d %d %d", inf.name, inf.mark[0],
inf.mark[1], inf.mark[2]);
  }
  fclose(f);
  return 0;
}
```

Иногда необходимо определить позицию указателя. Для этого можно воспользоваться функцией ***ftell()***:

```
long ftell(FILE *stream);
```

Возвращает значение указателя на текущую позицию файла. В случае ошибки возвращает число (-1).

int fsetpos(FILE *stream, **const long** *pos) - устанавливает значение указателя чтения-записи (указатель на текущую позицию) файла в позицию заданную значением по указателю **pos**. Возвращает нуль при корректном выполнении, и любое не нулевое значение при ошибке.

int fgetpos(FILE *stream, **long** *pos) - помещает в переменную, на которую указывает **pos**, значение указателя на текущую позицию в файле. Возвращает нуль при корректном выполнении, и любое не нулевое значение при ошибке. [3]

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

Варианты заданий

Задача 1 - максимум 4 балла при сдаче на том же занятии, на котором задача выдана. При сдаче на следующих занятиях максимум - 2 балла.

Задача 2 - максимум 4-5 баллов при сдаче на любом занятии. Принимается частичное решение, если это позволяет условие задачи.

Задача 1 должна быть сдана обязательно.

Вариант	Задание
1	1. Дан бинарный файл F1, содержащий целые числа. Дан текстовый файл F2, содержащий целые числа, разделенные пробелами. Удалить из F1 числа, содержащиеся в F2. Имена файлов получить из командной строки. Дополнительных коллекций не использовать. 2. В бинарном файле F1 содержится набор структур с информацией о других файлах (структура содержит путь к файлу и размер). При добавлении элемента в F1 создается пустой файл по заданному пути. Информация о файлах должна обновляться по запросу с клавиатуры: при удалении файла с диска соответствующая запись должна удаляться из F1, при изменении размера файла запись в F1 должна отобразить актуальный размер.

2	1. Имеются два бинарных файла, содержащих целые числа, упорядоченные по возрастанию. Переписать информацию из обоих файлов в третий файл, упорядоченный по возрастанию. Дополнительных коллекций и файлов не использовать. 2. В бинарном файле хранятся структуры, описывающие страну и содержащие кол-во населения, площадь, название. Вычислить и вывести размер файла. Дать возможность задать с клавиатуры максимальный размер файла, после чего файл должен уменьшиться, если этот размер превышен. После этого должна быть возможность вывести оставшиеся элементы. Дать возможность задать с клавиатуры площадь и удалить все страны с площадью меньше заданной. Дополнительных коллекций не использовать.
3	1. В бинарном файле указан размер квадратной матрицы, а затем построчно записаны элементы матрицы целых чисел. Таких групп данных в файле несколько. Для каждой группы вычислить сумму элементов, расположенных на главной диагонали. Результаты записать в новый текстовый файл. Имена файлов задаются в командной строке. Можно использовать дополнительную коллекцию для хранения одной матрицы. 2. Даны два файла. В файле F1 хранятся записи, представляющие из себя товары: уникальный идентификатор и название. В файле F2 хранятся магазины. У магазина есть название и набор идентификаторов товаров. Сделать возможность при выборе магазина выводить названия товаров, получаемых из F1 по идентификатору. Сделать возможность добавлять товары в магазины, удалять товары из F1.
4	1. Для существующего упорядоченного бинарного файла добавить возможность ввести с клавиатуры число. При этом число размещается в файле без нарушения его упорядоченности по возрастанию. Дополнительных коллекций и файлов не использовать. 2. В бинарном файле хранится список спортсменов: спринтеров, прыгунов в высоту, тяжелоатлетов. Все записи включают фамилию и страну. Спринтеры дополнительно содержат время финиша, прыгуны - взятую высоту и рост, тяжелоатлеты - взятый и собственный вес. В файле записи отсортированы по виду спорта и по результату. Добавить возможность добавлять записи в файл с учетом вида спорта и результата (у спринтеров первая запись - с наименьшим временем, у прыгунов - с наибольшей высотой и т.д.). Добавить возможность вывести файл. Дополнительных коллекций не использовать.

5	1. Из командной строки получить имя бинарного файла и границы диапазона чисел. Удалить из файла числа, находящиеся в заданном диапазоне. Дополнительных коллекций и файлов не использовать. Пример. Файл содержит числа 4, 1, 3, 22. Диапазон - от 1 до 3. Результат - 4, 22. 2. В бинарном файле хранится список студентов. У каждой записи есть ФИО и порядковый номер. ФИО может быть любой длины, т.е. размер в файле строка занимает не больше места, чем требуется для хранения конкретной строки. Реализовать добавление студентов в конец и удаление студентов по порядковому номеру с последующей актуализацией всех порядковых номеров. Дополнительных коллекций и файлов не использовать.
6	1. В бинарном файле хранится множество геометрических фигур (3 вида на выбор), каждая представлена своей структурой. Реализовать удаление всех фигур по заданному с клавиатуры типу, вывод содержимого файла на экран. Дополнительных коллекций и файлов не использовать. 2. В бинарном файле хранятся имена и количество детей, названных этим именем. Добавить возможность отсортировать данные в файле по убыванию количества и по алфавиту относительно имен. Добавить возможность вывода данных из файла. Дополнительных коллекций и файлов не использовать
7	1. В бинарном файле F1 хранятся структуры, хранящие операцию (в виде перечисления) и два операнда (числа с плавающей запятой). С клавиатуры ввести индекс элемента, удалить этот элемент из файла, выполнить хранящуюся в нем операцию, а результат дописать в текстовый файл F2. Дополнительных коллекций и файлов не использовать. 2. В бинарном файле хранятся данные, представленные целочисленным ключом и строковым значением. Строка может иметь любую длину, в файле должна занимать только необходимое ей место. Сделать возможность вывести все данные из файла, добавить запись, прочитать строку по ключу. При чтении запись переносится в начало файла. Новые данные также записываются в начало файла. Дополнительных коллекций и файлов не использовать.

8	1. В бинарном файле записаны целочисленные значения. Сделать циклический сдвиг элементов, записанных в файле, на один элемент вправо. Вывести результат на экран. Дополнительных коллекций и файлов не использовать. Пример. Изначально файл содержит числа 1, 2, 3, 4, 5, 6. В результате файл содержит числа 6, 1, 2, 3, 4, 5. 2. В бинарном файле F1 записана одна структура, в которой хранится состояние системы: флаг успешности последнего запуска, язык, дата последнего запуска, кол-во свободного места на диске, объем диска, версия системы, дата обновления. Вывести размер файла и все данные, находящиеся в нем. Задать максимальный размер файла. После этого разбить файл F1 на несколько файлов так, чтобы размер каждого файла не превышал максимальный. По запросу с клавиатуры отменить ограничения по максимальному размеру, объединить созданные файлы в исходный, удалить созданные файлы.
9	1. В бинарном файле хранятся пути, заданные наборами точек. По запросу с клавиатуры добавить новый путь. По запросу с клавиатуры удалить наибольший путь. Дополнительных коллекций и файлов не использовать. 2. В бинарном файле записаны структуры, хранящие фамилию студента, средний балл, номер группы, номер паспорта, упорядоченные по фамилии. С клавиатуры задать фамилию студента, найти его в файле бинарным поиском, вывести средний балл. Реализовать возможность добавить новую запись без нарушения упорядоченности данных, удалить запись по фамилии. Дополнительных коллекций и файлов не использовать.
10	1. В бинарном файле хранятся описания иконок, у которых есть ширина, высота в пикселях и путь к файлу изображения. По запросу с клавиатуры вывести все данные из файла либо удалить из файла все описания иконок, у которых по хранимому пути файлы отсутствуют. Дополнительных коллекций и файлов не использовать. 2. В бинарном файле F1 хранятся статьи, у которых есть название, кол-во страниц, массив названий книг, материалы из которых использованы в статьях. В бинарном файле F2 хранятся книги, у которых есть название, кол-во страниц и кол-во ссылок из статей. При добавлении статьи для всех книг, материалы из которых используются в статье, увеличивается кол-во ссылок. Реализовать добавление и удаление статьи. Когда на какую-то из книг не остается ссылок, она автоматически должна удалиться из файла F2. Дополнительных коллекций и файлов не использовать.

11	1. В бинарном файле хранятся товары, у которых есть название, цена, оценка и путь к текстовому файлу с детальным описанием. Каждому товару соответствует отдельный файл с детальным описанием. Вывести всю информацию (включая детальное описание) о товарах с оценкой больше M. Число M ввести с клавиатуры. Дополнительных коллекций не использовать. 2. В бинарном файле записана информация о населении, запись хранит ФИО, идентификационный номер, дату рождения, смещения в файле, по которому лежит информация о родителях. Заранее сформировать и вывести файл с исходными данными. С клавиатуры задать идентификационный номер, число N. Вывести данные о человеке и его предках до N-го уровня дальности (например, для N = 2 вывести информацию о матери, отце, матери и отце матери, матери и отце отца). Реализовать возможность удаления записи.
12	1. Дан файл F1, содержащий целые числа, упорядоченные по возрастанию и файл F2, содержащий целые числа, упорядоченные по убыванию. Переписать информацию из обоих файлов в третий файл, упорядоченный по убыванию. Дополнительных коллекций и файлов не использовать, сортировку не выполнять. 2. В бинарном файле F1 хранятся записи о товарах, у которых есть название, рейтинг, оценка и смещение в текстовом файле F2, содержащем детальные описания товаров, разделенные новыми строками. Добавить возможность вывести данные о товаре. Добавить возможность удалить товар, при этом его детальное описание должно удалиться из файла с детальными описаниями. Дополнительных коллекций и файлов не использовать.

13	1. В бинарном файле хранятся несколько изображений. Изображение представлено следующими данными: целочисленный ключ, количество байт в пикселе (3 или 4), ширина, высота, массив пикселей. Реализовать чтение и вывод массива пикселей по ключу, добавление изображения. 2. В бинарном файле F1 хранятся записи о мобильных устройствах: строка модели, ширина и высота экрана. В бинарном файле F2 хранятся другие записи о мобильных устройствах: строка модели, операционная система, плотность экрана. На основании файлов F1 и F2 создать файл F3, который хранит записи, содержащие модель, ширину и высоту экрана, операционную систему, плотность экрана. Если модели, указанной в одном из файлов, нет в другом из файлов, данные в F3 не добавлять, оставить их в исходном файле. Если модель есть в обоих файлах, удалить данные для этой модели из F1 и F2. Дополнительных коллекций не использовать.

14	1. В бинарном файле F1 хранится структура, представляющая собой HTTP-ответ. Структура содержит код ответа (структура, в которой находится целочисленный код и строковое описание кода), массив заголовков (структур, содержащих имя заголовка и его значение) и тело ответа (строка). Добавить возможность ввести данные в файл F1. Добавить возможность прочитать эти данные и создать текстовый файл F2, где на первой строке через пробел идет целочисленный код ответа, затем через пробел его строковое описание, на следующих строках для каждого заголовка имя, двоеточие, пробел, значение заголовка. Затем пустая строка и тело ответа. Добавить возможность без чтения структуры в память изменить каждое из полей HTTP-ответа в файле F1. Пример текста в F2: 200 Ok Content-Length: 19 Content-Type: text/html; charset=utf-8 Hello world! 2. В бинарном файле F1 хранятся записи, содержащие информацию о сбое (аварийной остановке) приложения. Записи содержат версию ОС, название ОС, модель процессора, IP-адрес, кол-во свободной памяти на диске, название приложения, дату, строку логов. Заранее создать файл с массивом сбоев. По запросу с клавиатуры вывести всю эту информацию на экран, добавить новые элементы в файл. По запросу с клавиатуры сформировать файл F2, который реорганизует данные из F1 следующий формат. Структура окружения содержит версию ОС, название ОС, модель процессора, IP-адрес и массив структур сбоя с одинаковым окружением, которая содержит кол-во свободной памяти на диске, название приложения, дату, строку логов. По запросу с клавиатуры вывести данные из F2 на экран. Пример. Данные в F1: https://jsoneditoronline.org/#left=cloud.1e6455bb9acb4b6eb756f742b0e2e0fd. Данные в F2: https://jsoneditoronline.org/#left=cloud.2ef98a984bd342059db2b88f95a0d754.

15	1. В бинарном файле F1 содержатся целочисленные значения. По запросу с клавиатуры на основании F1 записать бинарный файл F2, в который попадают числа из F1, встречающиеся больше одного раза. При этом в F2 необходимо записать структуры, содержащие целочисленное значение из F1 и кол-во их повторений в файле. При переносе в файл F2 из файла F1 соответствующие числа должны быть удалены (этот пункт можно доделать отдельно). Дополнительных коллекций не использовать. 2. В бинарном файле F1 хранятся целочисленные значения. Сформировать из них структуры интервалов, хранящих начальное и конечное значение для непрерывных последовательностей чисел и записать в эти структуры в новый бинарный файл F2. Пример. Файл F1: 7, 6, 2, 3, 8, 1. Файл F2: [1,3],[6,7],[8,8].