Лабораторная работа 9

Цель работы

Изучение принципов работы с двоичными файлами.

Задание

Разработать программу и подпрограмму (подпрограммы), работающую с двоичным файлом и выполняющую действия согласно варианту задания

№	Задание
1	Преобразовать входной текстовый файл в выходной двоичный, содержащий записи следующего вида: данные типа int – длина строки; массив типа char – строка входного файла без завершающего символа '\n'.
2	Преобразовать входной текстовый файл в выходной двоичный, содержащий записи следующего вида: данные типа int – длина строки; массив типа char фиксированной размерности, равной длине максимальной строки,– строка входного файла без завершающего символа '\n'. До максимальной длины массив типа char дополняется байтами со значением 0.
3	Преобразовать входной текстовый файл в выходной двоичный, содержащий следующие данные: значение типа double – количество строк в файле; n массивов типа char, каждый из которых содержит строку входного файла без завершающего символа '\n' и заканчивается нулевым байтом.
4	Преобразовать входной текстовый файл в выходной двоичный, содержащий записи следующего вида: данные типа float – длина строки; данные типа int – количество слов в строке; массив типа char – строка входного файла без завершающего символа '\n'.
5	Преобразовать входной текстовый файл в выходной двоичный, содержащий данные следующего вида: значение типа int – количество строк в файле (n), n массивов типа char фиксированной размерности, равной длине максимальной строки, каждый элемент содержит строку без завершающего символа '\n'. До максимальной длины массив типа char дополняется пробелами.
6	Преобразовать входной текстовый файл в выходной двоичный, содержащий записи следующего вида: данные типа char – длина строки; данные типа char – количество букв в строке; данные типа int – сумма кодов символов строки; массив типа char – строка входного файла без завершающего символа '\n'.

ПРИМЕЧАНИЕ. В каждом задании после формирования выходного двоичного файла необходимо вывести на экран содержимое и размер полученного двоичного файла.

При выполнении работы необходимо разработать две программы: первая преобразовывает входной тестовый файл в выходной двоичный, а вторая – распечатывает содержимое двоичного файла заданной структуры и его размер в байтах.

Теоретическая часть

Ввод-вывод для двоичных файлов

Для выполнения операций ввода-вывода для двоичных файлов:

- **int fread(void*ptr,int size, int n, FILE*f)**, где void*ptr – указатель на область памяти, в которой размещаются считанные из файла данные, int size – размер одного считываемого элемента, int n – количество считываемых элементов, FILE*f – указатель на файл, из которого производится считывание. В случае успешного считывания функция возвращает количество считанных элементов, иначе – отрицательное значение.
- **int fwrite(void*ptr,int size, int n, FILE*f)**, где void*ptr – указатель на область памяти, в которой размещаются считанные из файла данные, int size – размер одного записываемого элемента, int n – количество записываемых элементов, FILE*f – указатель на файл, в который производится запись. В случае успешной записи функция возвращает количество записанных элементов, иначе – отрицательное значение.

Прямой доступ к файлам

Рассмотренные ранее функции обмена с файлами позволяют записывать и считывать данные только последовательно. Операции чтения/записи всегда производятся, начиная с текущей позиции в потоке. Начальная позиция устанавливается при открытии потока и может соответствовать начальному или конечному байту потока в зависимости от режима открытия файла. При открытии потока в режимах "r" и "w" указатель текущей позиции устанавливается на начальный байт потока, при открытии в режиме "a" - за последним байтом в конец файла. При выполнении каждой операции указатель перемещается на новую текущую позицию в соответствии с числом записанных/прочитанных байтов.

Средства прямого доступа дают возможность перемещать указатель текущей позиции в потоке на нужный байт. Для этого используется функция

```
int fseek(FILE *f, long off, int org), где
```

FILE *f - – указатель на файл,
long off – позиция смещения
int org – начало отсчета. Начало отсчета задается с помощью одной из определенных в файле stdio.h констант:
SEEK_SET – начало файла;
SEEK_CUR – текущая позиция;
SEEK_END – конец файла.

Функция fseek возвращает 0, если перемещение в потоке выполнено успешно, иначе возвращает ненулевое значение.

Кроме этой функции, для прямого доступа к файлу используются:

```
long ftell(FILE *f);//получает значение указателя текущей позиции в потоке;
void rewind(FILE *f);//установить значение указателя на начало потока.
```

Рекомендации по выполнению лабораторной работы

Двоичный файл – это упорядоченная по определенным правилам последовательность двоичных данных: байт, целых чисел, структур и т.д. Поэтому прежде чем считывать данные из этого файла, нужно узнать его структуру – только после этого можно буть уверенным в том, что данные будут считаны правильно.

Данные из двоичного файла считываются блоками, причем блоки содержат именно те единицы информации, которые нужно, - целые числа, вещественные числа или что-нибудь другое. При считывании информации из двоичных файлов можно различными способами организовывать циклы чтения информации, но использование функции feof() в качестве параметра цикла не всегда является удобным, поскольку признак конца файла устанавливается после попытки чтения за концом файла и никак не раньше.

Пример неверного использования функции feof.

Предположим, нужно считать двоичную информацию – байты из некоторого файла file.bin, размер которого заранее неизвестен. Предположим, что у нас для решения этой задачи написана следующая программа:

```
#include <stdio.h>

void main()
{
char buf[256];                    /* массив для считываемых байт, но не символов! */
FILE *in;
     in = fopen("file.bin","rb");
     if(in != NULL)
     {
          while(feof(in) == 0)    /* читаем до конца файла */
          {
               fread(buf,1,256,in);    /* прочитаем 256 байт из файла */
          /*
          здесь производится обработка считанной информации
          */
          }
          fclose(in);             /* закроем файл */
     }
}
```

Предположим что в файле file.bin количество байт кратно 256, например, точно 256. В этом случае программа будет выполняться слкдующим образм:

Первое выполнение цикла while:

- функция feof возвращает 0, поскольку файл не закончился;
- функция fread читает из файла 256 байт и признак конца файла не устанавливается, потому что из файла считано столько единиц информации, сколько было запрошено;

Второе выполнение цикла while:

- функция feof возвращает 0, поскольку файл еще не закончился, несмотря на то, что из него считаны все 256 байт;
- функция fread считает из файла 0 байт – файл закончился, содержимое буфера buf не изменяется, и только теперь устанавливается признак конца файла;

Третье выполнение цикла while:

- функция feof возвращает не нулевое значение и цикл завершается.

Таким образом, в программе цикл будет выполнен один лишний раз, хотя если во входном файле количество будет не кратно 256, то программа будет работать корректно.

Приведем пример программы, которая при тех же условиях будет корректно работать при люом размере входного файла.

```
#include <stdio.h>

void main()
{
char buf[256];                    /* массив для считываемых байт, но не символов! */
FILE *in;
size_t n_obj;
     in = fopen("file.bin","rb");
     if(in != NULL)
     {
          while((n_obj=fread(buf,1,256,in)) != 0)   /* читаем до конца файла */
          {
          /*
          здесь производится обработка считанной информации
          n_obj – количество действительно прочитанных
                  единиц информации
          */
          }
          fclose(in);             /* закроем файл */
     }
}
```

Содержание отчета

Отчет по лабораторной работе должен содержать:

- задание лабораторной работы, соответствующее варианту
- структурную схему алгоритма программы и подпрограммы (подпрограмм)
- текст программы
- результаты работы программы