Министерство образования Республики Беларусь

Учреждение образования

«Белорусский государственный университет

информатики и радиоэлектроники»

Кафедра систем управления

Отчет по лабораторной работе № 9

Тема: «Программирование c использованием файлов»

Вариант 7

Выполнил:

студент группы 322441 Калинин В.А.

Проверил:

зам.каф ИТАС Навроцкий А.А.

Минск

2023

Цель работы

Цель работы: изучить правила создания и обработки данных структурного типа с использованием файлов, написать и отладить программу по созданию файлов.

**Краткие теоретические сведения**

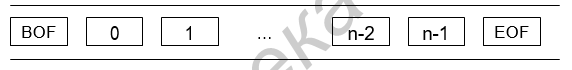
Файл − поименованная совокупность данных, расположенных на внешнем носителе. В начале работы для получения доступа к данным файл необходимо открыть. При открытии файла указатель текущей позиции помещается в начало файла. После выполнения любой операции над данными указатель сдвигается на одну позицию вперед. В конце работы файл закрывается, т. е. доступ к данным, размещенным в файле, будет запрещен. Информация о файле хранится в управляющей структуре, имеющей тип FILE.

Различают два вида файлов: текстовые и двоичные.

Текстовые файлы хранят информацию в виде последовательности символов. Вывод осуществляется аналогично выводу на экран. Текстовые файлы могут быть отредактированы в любом текстовом редакторе.

Бинарные (или двоичные) файлы предназначены для хранения последовательности байтов. Структура такого файла определяется программно.

Файлы, размещаемые на носителях информации, имеют следующую структуру:



В начале файла записана информация о файле BOF (Begin of File), его имя, тип, длина и т. д., в конце файла помещается признак конца файла EOF (End of File). Если файл пуст, то BOF и EOF совмещены.

При работе с файлами используются следующие макросы:

* NULL − определяет пустой указатель;
* EOF − значение, возвращаемое при попытке чтения после конца файла;
* FOPEN\_MAX − возвращает максимальное число одновременно открытых файлов.

**Функции для работы с файлами**

Функции для работы с файлами размещены в библиотеках stdio.lib и io.lib. При работе с файлами используются указатели типа FILE. Формат объявления указателя на файл, следующий:

FILE \*указатель\_на\_файл;

Например:

FILE \*fl1, \*fl2;

Указатель содержит адрес структуры, включающей в себя различные сведения о файле, например, его имя, статус и указатель на начало файла.

Функция

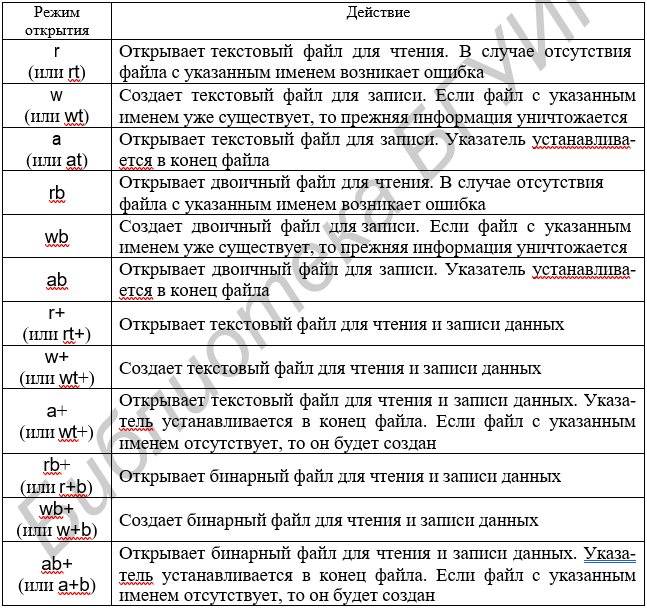
FILE \*fopen(const char \*имя\_файла,

const char \*режим\_открытия); открывает файл и связывает его с потоком. Возвращает указатель на открытый файл.

Имя\_файла − указатель на строку символов, в которой хранится имя

файла и путь к нему. Например: “d:\\work\\lab2.dat”.

Режим\_открытия − указатель на строку символов, в которой указывается режим открытия файла.

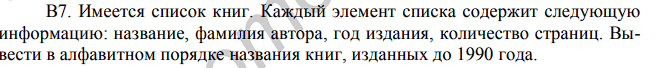


По умолчанию файл открывается в текстовом режиме.

Если при открытии файла произошла ошибка, функция fopen возвращает значение NULL.

Задание

Задание: написать набор функций для выполнения следующих задач: создание бинарного файла; запись данных в файл; открытие файла и чтение из него данных; вывод результата на экран; вывод результата в текстовый файл. Для вызова необходимых функций использовать меню.



Листинг программы

#include <iostream>

#include <fstream>

#include <cstring>

#include <algorithm>

using namespace std;

struct Book {

char title[100];

char author[100];

int year;

int pages;

};

bool compareBooks(const Book& book1, const Book& book2) {

return strcmp(book1.title, book2.title) < 0;

}

void createBinaryFile(const char\* fileName, const Book\* books, int n) {

ofstream file(fileName, ios::binary | ios::trunc);

if (file.is\_open()) {

file.write(reinterpret\_cast<const char\*>(&n), sizeof(int));

file.write(reinterpret\_cast<const char\*>(books), n \* sizeof(Book));

file.close();

cout << "Binary file created successfully.\n";

} else {

cerr << "Error opening file for writing.\n";

}

}

void readBinaryFileAndPrint(const char\* fileName) {

ifstream file(fileName, ios::binary);

if (file.is\_open()) {

int n;

file.read(reinterpret\_cast<char\*>(&n), sizeof(int));

Book\* books = new Book[n];

file.read(reinterpret\_cast<char\*>(books), n \* sizeof(Book));

sort(books, books + n, compareBooks);

cout << "\nBooks published before 1990 in alphabetical order:\n";

for (int i = 0; i < n; ++i) {

if (books[i].year < 1990) {

cout << "Title: " << books[i].title << ", Author: " << books[i].author

<< ", Year of publication: " << books[i].year << ", Number of pages: " << books[i].pages << endl;

}

}

delete[] books;

file.close();

} else {

cerr << "Error opening file for reading.\n";

}

}

void writeToFile(const char\* binaryFileName, const char\* textFileName) {

ifstream binaryFile(binaryFileName, ios::binary);

ofstream textFile(textFileName);

if (binaryFile.is\_open() && textFile.is\_open()) {

int n;

binaryFile.read(reinterpret\_cast<char\*>(&n), sizeof(int));

Book\* books = new Book[n];

binaryFile.read(reinterpret\_cast<char\*>(books), n \* sizeof(Book));

sort(books, books + n, compareBooks);

textFile << "\nBooks published before 1990 in alphabetical order:\n";

for (int i = 0; i < n; ++i) {

if (books[i].year < 1990) {

textFile << "Title: " << books[i].title << ", Author: " << books[i].author

<< ", Year of publication: " << books[i].year << ", Number of pages: " << books[i].pages << endl;

}

}

delete[] books;

binaryFile.close();

textFile.close();

cout << "Data successfully written to the text file.\n";

} else {

cerr << "Error opening files for writing.\n";

}

}

int main() {

setlocale(LC\_ALL, "");

int choice;

const char\* binaryFileName = "books.dat";

const char\* textFileName = "result.txt";

do {

cout << "\nMenu:\n";

cout << "1. Create a binary file with book data\n";

cout << "2. Display book data from the binary file on the screen\n";

cout << "3. Write book data from the binary file to a text file\n";

cout << "0. Exit\n";

cout << "Choose an action: ";

cin >> choice;

switch (choice) {

case 1: {

int n;

cout << "Enter the number of books: ";

cin >> n;

Book\* books = new Book[n];

for (int i = 0; i < n; ++i) {

cout << "Enter information about book #" << i + 1 << ":\n";

cout << "Title: ";

cin.ignore();

cin.getline(books[i].title, sizeof(books[i].title));

cout << "Author: ";

cin.getline(books[i].author, sizeof(books[i].author));

cout << "Year of publication: ";

cin >> books[i].year;

cout << "Number of pages: ";

cin >> books[i].pages;

}

createBinaryFile(binaryFileName, books, n);

delete[] books;

break;

}

case 2:

readBinaryFileAndPrint(binaryFileName);

break;

case 3:

writeToFile(binaryFileName, textFileName);

break;

case 0:

cout << "Program terminated.\n";

break;

default:

cout << "Invalid choice. Please choose an existing menu item.\n";

break;

}

} while (choice != 0);

return 0;

}

Результат работы программы