**Лабораторная работа №8**

**Работа с файлами**

**1. Функция, читающая строки из клавиатуры и записывающая их в файл**

#include <iostream>

#include <fstream>

using namespace std;

// Функция читает строки из клавиатуры и записывает их в файл

void Example1(const char\* filename)

{

ofstream os(filename); // текстовый файл для вывода

// проверка, открыт ли файл

if (!os)

{

cout << "Cannot open the file to output. \n";

return;

}

char str[80];

cout << "Save the lines to HDD. Enter ! to stop." << endl;

do

{

cout << ":";

// чтение строки с использоанием пробелов между словами

cin.getline(str, 80);

os << str << endl; // записать в файл строку str

} while (\*str != '!'); // конец ввода строк - !

os.close(); // закрыть файл

return;

}

void main()

{

Example1("myfile.txt"); }

После вызова функции Example1() будет создан файл «myfile.txt», в котором будут записаны строки, введенные с клавиатуры.

Результат работи программы:

Save the lines to HDD. Enter ! to stop.

:Hello world!

:bestprog.net

:Working with files in C++

:!

**2. Функция, которая читает текстовый файл и выводит его содержимое на экран**

Функция Example2() читает содержимое файла filename, имя которого есть входящим параметром функции.

#include <iostream>

#include <fstream>

using namespace std;

// Функция, которая читает файл filename и выводит его на экран

// Используется класс ifstream для чтения файла

bool Example2(const char \* filename)

{

// создать экземпляр класса

ifstream is(filename); // открыть файл в конструкторе

// если файл не открыт, то выход с кодом false

if (!is)

return false;

// чтение строк в цикле while

char str[100]; // буфер для чтения одной строки

while (is) // если is ненулевое, то еще не конец файла

{

is.getline(str, 100); // прочитать строку из файла в буфер str

cout << str << endl; // вывести на екран

}

is.close(); // закрыть файл

return true;

}

void main()

{

Example2("myfile.txt");

}

Результат работы программы отображает содержимое файла «myfile.txt», созданного в п.1 этой темы

Hello world!

bestprog.net

Working with files in C++

!

**3. Пример бесформатного ввода/вывода. Копирование одного файла в другой**

В примере реализована функция Example3(), которая выполняет копирование файлов в двоичном (бинарном) формате. Функция получает два параметра. Первый параметр типа const char\* есть имя файла-источника. Второй параметр типа const char\* есть имя файла-назначения.  
Функция реализует посимвольное копирование. Для получения символа из файла-источника используется функция get().

#include <iostream>

#include <fstream>

using namespace std;

// Бесформатный ввод/вывод, копирование одного файла в другой

// Функция читает файл readFile и записывает его в файл writeFile

// использует функцию get() для чтения

bool Example3(const char \* readFile, const char \* writeFile)

{

// создать экземпляры классов ifstream, ofstream

ifstream rf(readFile, ios::in | ios::binary); // файл-источник (чтение)

// проверка, открыт ли файл-источник

if (!rf)

{

cout << "Cannot open source file." << endl;

return false;

}

// файл-назначение (запись)

ofstream wf(writeFile, ios::out | ios::binary);

// проверка, открыт ли файл-назначение

if (!wf)

{

rf.close(); // закрыть файл-источник

cout << "Cannot open the files." << endl;

return false;

}

char sym;

// цикл посимвольного чтения

while (rf)

{

rf.get(sym); // считать из rf => sym

if (rf)

wf << sym; // записать sym => wf

}

cout << "Copy result: OK!" << endl;

rf.close(); // закрыть оба файла

wf.close();

}

Вызов функции Example3() из функции main() может быть следующим:

void main()

{

Example3("myfile.txt", "myfile2.txt");

}

После выполнения функции Example3() будет создан файл «myfile2.txt», который будет копией файла «myfile.txt».

**4. Пример бесформатного ввода/вывода. Копирование одного файла в другой. Функция put()**

Функция Example4() из данного примера работает также как и предыдущая, только вместо вывода в поток **<<** используется функция put(). Также, с помощью функции is\_open() выполняется проверка на корректность открытия файла.

#include <iostream>

#include <fstream>

using namespace std;

// То же, что и Example3(), только для записи использует функцию put()

bool Example4(const char \* readFile, const char \* writeFile)

{

ifstream rf(readFile, ios::in | ios::binary);

// проверка, открыт ли файл

if (!rf.is\_open())

return false;

ofstream wf(writeFile, ios::out | ios::binary);

// проверка, открыт ли файл

if (!wf.is\_open())

{

rf.close(); // закрыть ранее открытый файл rf

return false;

}

char sym;

// цикл посимвольного чтения

while (rf)

{

rf.get(sym); // считать из rf => sym

if (rf) wf.put(sym); // записать sym => wf

}

rf.close(); // закрыть оба файла

wf.close();

}

void main()

{

Example4("myfile.txt", "myfile2.txt");

}

**5. Пример функции записывающей структурную переменную в файл**

По данному примеру можно реализовывать собственные функции, которые будут записывать структуры или классы в файл.

Реализована функция Example5(), которая выполняет запись структурной переменной типа BOOK в файл, имя которого есть входящим параметром. Функция Example5() использует функцию write() для записи. Файл открывается в двоичном формате (ios::binary).

#include <iostream>

#include <fstream>

#include <string.h>

using namespace std;

// Функция, которая записывает структуру типа BOOK в файл,

// использует функцию write()

struct BOOK

{

char title[100]; // название книги

char author[70]; // автор

int year; // год выпуска

float price; // стоимость книги

};

bool Example5(const char \* filename)

{

// создать структуру

BOOK B;

strcpy\_s(B.title, "Title of book");

strcpy\_s(B.author, "Authof of book");

B.year = 2000;

B.price = 12.65f;

// создать экземпляр файла в двоичном формате (ios::binary)

ofstream outFile(filename, ios::out | ios::binary);

if (!outFile) return false;

// запись структурной переменной B в файл

outFile.write((char\*)&B, sizeof(BOOK));

outFile.close(); // закрыть файл

return true;

}

void main()

{

Example5("myfile3.txt");

}

**6. Пример чтения структурной переменной из файла**

Данный пример есть продолжением предыдущего примера из пункта 5. В примере в функции Example6() заполняется значение структурной переменной типа BOOK. Полученное значение формируется как входящий параметр-ссылка на тип BOOK. Также функция получает параметром имя файла для чтения. Для чтения структурной переменной используется функция read().

#include <iostream>

#include <fstream>

#include <string.h>

using namespace std;

// чтение структуры типа BOOK из файла filename

// используется функция read

bool Example6(const char \* filename, BOOK& B)

{

ifstream inFile(filename, ios::in | ios::binary);

if (!inFile) return false;

// получить структурную переменную B из файла

inFile.read((char \*)&B, sizeof(BOOK));

inFile.close();

return true;

}

void main()

{

BOOK B; // чтение из файла і запись в структурную переменную B

Example5("myfile3.txt"); // запись в файл

Example6("myfile3.txt", B); // чтение из файла

// вывести содержимое переменной B

cout << "B.title = " << B.title << endl;

cout << "B.author = " << B.author << endl;

cout << "B.price = " << B.price << endl;

cout << "B.year = " << B.year << endl;

}

Результат работы программы

B.title = Title of book

B.author = Authof of book

B.price = 12.65

B.year = 2000

**7. Пример чтения/записи массива структур в файл. Функции write(), read()**

В примере используются функции write(), read() для работы со структурой типа BOOK, а именно:

* записи массива типа BOOK в файл, который состоит из трех структурных переменных;
* чтения массива структур типа BOOK из файла.

#include <iostream>

#include <fstream>

#include <string>

using namespace std;

// Структура BOOK

struct BOOK

{

char title[100]; // название книги

char author[70]; // автор

int year; // год выпуска

float price; // стоимость книги

};

// Запись массива структур в файл с помощью функции write()

// Чтение массива структур из файла с помощью функции read()

bool Example7(const char \* filename)

{

// демонстрируется запись в файл и чтение из этого файла массива структур типа BOOK

// создать массив структур

BOOK B[3] = {

{ "Title-01", "Author-01", 2005, 100.95 },

{ "Title-02", "Author-02", 2008, 90.25 },

{ "Title-03", "Author-03", 2002, 180.50 }

};

int n = 3; // количество элементов в массиве B

BOOK C[3]; // другой масив, в который будет осуществлено чтение

int n2; // количество элементов в массиве C

int i;

// 1. Запись массива структур в файл

// outF - экземпляр файла, в который реализуется запись

ofstream outF(filename, ios::out | ios::binary);

if (!outF) return false;

// записать значение n

outF.write((char\*)&n, sizeof(int));

// запись массива B[] в файл wf

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

outF.write((char\*)&(B[i]), sizeof(BOOK));

cout << "Array is written\n" << endl;

// после завершения работы с файлом его нужно закрыть (обязательно)

outF.close();

// 2. Чтение массива структур из файла

// inF - экземпляр файла, из которого происходит чтение

ifstream inF(filename, ios::in | ios::binary);

if (!inF) return false;

cout << "Read the array...\n";

// Сначала прочитать количество записанных структур

inF.read((char\*)&n2, sizeof(int));

// цикл чтения массива структур в переменную C

for (i = 0; i < n2; i++)

inF.read((char\*)&(C[i]), sizeof(BOOK));

inF.close(); // закрыть файл

// вывод массива C на экран

cout << "Array C:" << endl;

for (i = 0; i < n2; i++)

{

cout << "Title = " << C[i].title << ", ";

cout << "Author = " << C[i].author << ", ";

cout << "Year = " << C[i].year << ", ";

cout << "Price = " << C[i].price << endl;

}

}

void main()

{

Example7("file7.bin");

}

Результат работы программы

Array is written

Read the array...

Array C:

Title = Title-01, Author = Author-01, Year = 2005, Price = 100.95

Title = Title-02, Author = Author-02, Year = 2008, Price = 90.25

Title = Title-03, Author = Author-03, Year = 2002, Price = 180.5

**8. Пример записи/чтения массива чисел типа double**

Демонстрируется:

* запись в файл массива **M** чисел типа double функцией write();
* чтение из файла массива чисел типу double функцией read().

Файл открывается в двоичном формате.

#include <iostream>>

#include <fstream>

using namespace std;

// использование функций read(), write() для записи/чтения массива чисел

bool Example8(const char \* filename)

{

double M[] = { 2.44, 3.85, -3.23, 11.85, 3.38 }; // массив чисел

int i;

int n = 5; // количество элементов в массиве M

// 1. Запись массива в файл

// 1.1. Создать экземпляр outF, связанный с файлом filename

ofstream outF(filename, ios::out | ios::binary); // для записи, двоичный формат

// 1.2. Проверка открытия файла

if (!outF)

{

cout << "Error. Cannot open the file.";

return false;

}

// 1.3. Записать количество элементов в массиве M

outF.write((char\*)&n, sizeof(int));

// 1.4. Записать весь массив в файл

outF.write((char\*)&M, sizeof(double)\*n);

outF.close(); // закрыть файл

// 2. Чтение данных из файла filename в массив M2

double M2[5];

int n2;

// 2.1. Открыть файл для чтения

ifstream inF(filename, ios::in | ios::binary);

// 2.2. Проверка, открыт ли файл?

if (!inF)

{

cout << "Error. Cannot open file.";

return false;

}

// 2.3. Прочитать количество элементов в массиве

inF.read((char\*)&n2, sizeof(int));

// 2.4. Считать данные из файла в массив M2

inF.read((char\*)&M2, sizeof(double)\*n2);

inF.close(); // закрыть файл

// 2.5. Вывести массив M2 на экран

cout << "Array M2:\n";

for (i = 0; i < n2; i++)

cout << M2[i] << " ";

cout << endl;

return true;

}

void main()

{

Example8("file8.bin");

}

Результат работи программы:

Array M2:

2.44 3.85 -3.23 11.85 3.38

**9. Пример чтения из файла строк. Функция getline()**

В функции Example9() реализуется построчное чтение из файла с помощью функции getline(). Имя файла задается входящим параметром filename. Строки в файле состоят из набора слов, разделенных символом «пробел».

#include <iostream>

#include <fstream>

using namespace std;

// Чтение из файла. Функция getline()

bool Example9(const char\* filename)

{

ifstream inputFile(filename);

char buffer[255]; // буфер для сохранения одной строки

if (!inputFile)

{

cout << "Error. Cannot open file";

return false;

}

// цикл чтения строк из файла

while (inputFile)

{

inputFile.getline(buffer, 255);

if (inputFile)

cout << buffer << endl; // вывод прочитанной строки на экран

}

inputFile.close();

return true;

}

**10. Пример чтения строк из файла. Функции getline() + eof()**

В примере реализована функция Example10(), которая выполняет чтение строк из файла. Файл открывается в текстовом формате. Имя файла задается входящим параметром функции. Определение конца файла выполняется с помощью функции eof().

#include <iostream>

#include <fstream>

using namespace std;

// чтение из файла. Функции getline()+eof()

bool Example10(const char\* filename)

{

// создать экземпляр файла filename

ifstream inputFile(filename, ios::in);

if (!inputFile) return false;

char buffer[255]; // буфер для сохранения одной строки

// цикл чтения строк файла

// строки читаются до тех пор, пока не будет достигнут конец файла

while (!inputFile.eof())

{

inputFile.getline(buffer, 255);

if (inputFile)

cout << buffer << endl;

}

inputFile.close();

}