Міністерство освіти і науки України

Національний університет „Львівська політехніка”



**Звіт**

з лабораторної роботи №7

з дисципліни: «Програмування, частина 1»

Виконав:

студ.гр. КІ-17  
Жлуктенко Дмитро Миколайович

Прийняв:

Асистент Бойко Г. В.

Львів

2016

#### **Мета роботи:**

* познайомитися iз засобами файлового вводу та виводу в мовi програмування С.

#### **Теоретичний вступ:**

Файловий ввід та вивід

З точки зору програми на мові С всі типи файлів при виконанні операцій вводу та виводу розглядаються як потік логічних даних незалежно від того чи програма читає (записує) символ чи байт, чи рядок даних, чи блок даних заданого розміру. Потоки в С можуть бути або текстовими потоками або бінарними потоками. При відкриванні файла з допомогод функції fopen( ) створює новий потік, який існує до тих пір доки не буде закритий з допомогою функції fclose( ). В мові С виконання операцій вводу-виводу делегується виконавчому середовищу (операційній системі) у якому виконується програма. Таким чином потік є каналом через який дані можуть передаватися з виконавчого середовища в програму чи з програми у виконавче середовище. Операції вводу-виводу з пристроями виконуються таким самим чином як і з файлами.

Текстовий потік транспортує символи тексту, які складаються з рядків (розділені на рядки). Рядки тексту складаються з послідовності символів, які завершуються символом нового рядка (керуючим символом). Рядок може бути і порожнім.

Бінарний потік є послідовністю байтів, які жодим чином не інтерпретуються і передаються без модифікації. У бінарному потоці не виділяють жодних керуючих керуючих символів. Дані записані в бінарний потік можуть бути знову прочитані у такими ж як були записані.

Файли в С представляються послідовністю байтів. Функція fopen( ) зв'язує файл з потоком і ініціалізує об'єкт типу FILE, який містить всю необхідну інформацію для управління потоком. Ця інформація включає вказівник на використовуваний буфер, індикатор поточної позиції у файлі тоа інші.

Функція fopen( ) повертає вказівник на об'єкт FILE потоку, який пов'язаний з відкритим файлом. Після відкриття файла з ним можна виконувати функції, які виконують читання-запис даних. Ці функції мають вказівник на об'єкт FILE як один із аргументів.

Прототип стандартної бібліотечної функції fopen( ) наступний

FILE \*fopen( const char \* filename, const char \* mode );

Функція відкриває файл з іменем, який задається рядком filename. Параметр filename може містити і частину, яка специфікує каталог. Другий аргумент, mode, також є рядком і задає режим доступу. Цей аргумент вказує, які операції допустимі з файлом. Найважливішими є наступні значення цього аргументу:

"rb" - читання з бінарного потоку

"wb" - запис в бінарний потік

"rt" - читання з текстового потоку

"wt" - запис в текстовий потік

Для закривання фала використовується функція fclose( ). Її прототип:

int fclose( FILE \*fp );

Функція скидує дані, якщо вони ще залишилися в буфері в файл, закриває файл і звільняє пам'ять, яку займали буфери вводу-виводу. Функція fclose( ) повертає нуль при успішному завершенні чи EOF, якщо станеться помилка.

Для читання символів використовується функція

int fgetc( FILE \* fp );

Функція fgetc( ) читає символ з вхідного потоку на який посилається fp. Вона повертає прочитаний символ чи EOF, якщо станеться помилка.

Запис окремого символу в потік здійснюється з допомогою функції

int fputc( int c, FILE \*fp );

Функція fputc( ) записує символ, який задається аргументом c у вихідний потік на який посилається fp. Вона повертає записаний символ чи EOF, якщо станеться помилка.

Читання рядка з потоку виконується функцією

char \*fgets( char \*buf, int n, FILE \*fp );

Функція fgets( ) читає до n - 1 символів з вхідного потоку, на який посилається fp в буфер, який адресується buf, додаючи нуль-символ для завершення рядка. Якщо функція виявить символ нового рядка чи кінець файла пред тим як прочитає максимальну кількість символів, то лише вони будуть прочитані в буфер. Символ нового рядка '\n' також поміщається в буфер (якщо буде прочитаний). Функція повертає її аргумент buf чи нуль-вказівник при помилці чи якщо відстуні символи при досягненні кінця файла.

Запис рядка, що завершується нуль-символом в потік виконується функцією

int fputs( const char \*s, FILE \*fp );

fputs( ) записує рядок s у вихідний потік, на який посилається fp. Нуль-символ, який завершує рядок у вихідний потік не записується. Функція повертає EOF, якщо трапиться помилка, чи невід'ємне значення при успішному завершенні.

Функція fread( ) читає до n об'єктів, розмір яких size із вхідного потоку на який посилається fp і поміщає їх в масив buffer

size\_t fread( void \*buffer, size\_t size, size\_t n, FILE \*fp );

Функція повертає кількість прочитаних об'єктів. Якщо це значення є меншим за аргумент n, то це вказує, що був досягнутий кінець файла чи сталася помилка.

Функція fwrite( ) посилає n об'єктів розміром size з масиву buffer у вихідний потік на який посилається fp

size\_t fwrite( const void \*buffer, size\_t size, size\_t n, FILE \*fp );

Як і попередня функція вона повертає кількість записаних об'єктів. Якщо це значення є меншим за аргумент n, то це вказує, що сталася помилка.

З допомогою сімейства функцій printf( ) в мові С забезпечується форматований вивід. Для роботи з файлами використовується функція fprintf( )

int fprintf( FILE \* fp, const char \* format, ... );

Ця функція працює так само як і функція printf( ) за винятком того, що здійснює вивід у вихідний потік на який посилається fp.

Аналогічно, сімейство функцій scanf( ) служить для форматованого вводу. Для вводу з файлу служить функція fscanf( ).

int fscanf( FILE \* fp, const char \* format, ... );

Функція читає з вхідного потоку, на який посилається fp. Решта параметрів такі ж як і у функції scanf( ).

**Індивідуальне завдання №1. (Варіант 9)**

Індивідуальні завдання 8 ті ж самі що і [індивідуальні завдання 5](file:///C:\Users\judie\Downloads\lab\lab\zavdannia\Zavd5Lab5.pdf) чи [індивідуальні завдання 6](file:///C:\Users\judie\Downloads\lab\lab\zavdannia\Zavd6Lab5.html) (уточнити конкретне завдання у викладача), які виконувалися у лабораторній роботі №5. Відмінність полягає у тому, що ввід та вивід виконується через файли.

Небхідно написати два варіанти програми:

* програма у якій ввід здійснюється з текстового файлу і вивід виконується в текстовий файл;
* програма у якій ввід здійснюється з бінарного файлу і вивід відбувається в бінарний файл.

У бінарному файлі дані зберігаються у внутрішньому представлені.

Для варіанту, який використовує бінарні файли, необхідно написати дві допоміжні програми:

* допоміжну програму яка формує бінарний файл із вхідними даними;
* допоміжну програму яка читає вихідний бінарний файл і виводить на екран монітори вихідні дані.

Варіант 1:

#include <iostream>

#include <vector>

#include <fstream>

#include <string>

using namespace std;

void main()

{

vector<int> data;

int tmp = 0;

int max = 0;

int count = 0;

ifstream read("data.txt");

ofstream result("res.txt");

int lines = 0;

read >> lines; //number of lines

for (int i = 1; i <= lines; i++)

{

read >> tmp;

data.push\_back(tmp);

}

read.close();

cout << "File input:" << endl;

for (int i = 0; i < lines; i++)

{

cout << data[i] << endl;

}

for (int i = 0; i < lines; i++)

{

if (data[i] > max)

{

max = data[i];

count = 1;

}

else if (data[i] == max)

{

count++;

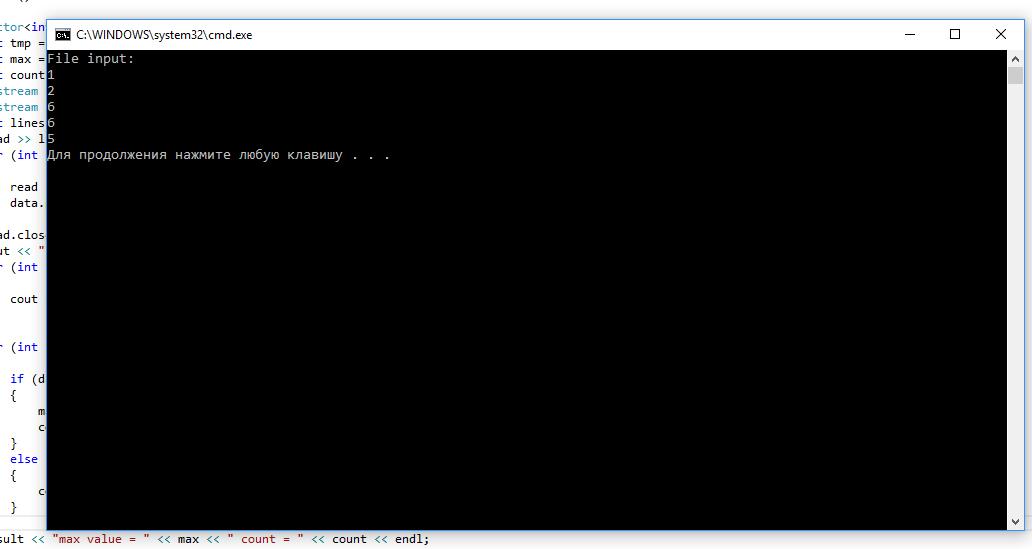
}

}

result << "max value = " << max << " count = " << count << endl;

result.close();

}



Варіант 2:

#include <iostream>

#include <vector>

#include <fstream>

#include <string>

#include <time.h>

using namespace std;

void writeToBinData(const char \* filename)

{

int lines = rand() % 10;

int tmp = 0;

ofstream iFile(filename, ios::binary);

iFile.write((char\*)&lines, sizeof(lines));

for (int i = 1; i <= lines; i++)

{

tmp = rand() % 100;

iFile.write((char\*)&tmp, sizeof(tmp));

}

iFile.close();

}

void readFromBinResults(const char \* filename)

{

int tmp1 = 0;

int tmp2 = 0;

ifstream i(filename, ios::binary);

i.read((char\*)&tmp1, sizeof(tmp1));

i.read((char\*)&tmp2, sizeof(tmp2));

cout << tmp1 << " " << tmp2 << endl;

}

void main()

{

// made without forming input data binary file !!!!!

// just cant get it

srand(time(NULL));

vector<int> data;

const char \* iFile = "data.txt";

const char \* oFile = "res.bin";

int tmp = 0;

int max = 0;

int count = 0;

ifstream read(iFile);

ofstream result(oFile, ios::binary);

int lines = 0;

read >> lines; //number of lines

for (int i = 1; i <= lines; i++)

{

read >> tmp;

data.push\_back(tmp);

}

read.close();

cout << "File input:" << endl;

for (int i = 0; i < lines; i++)

{

cout << data[i] << endl;

}

for (int i = 0; i < lines; i++)

{

if (data[i] > max)

{

max = data[i];

count = 1;

}

else if (data[i] == max)

{

count++;

}

}

result.write((char\*)&max, sizeof(max));

result.write((char\*)&count, sizeof(count));

result.close();

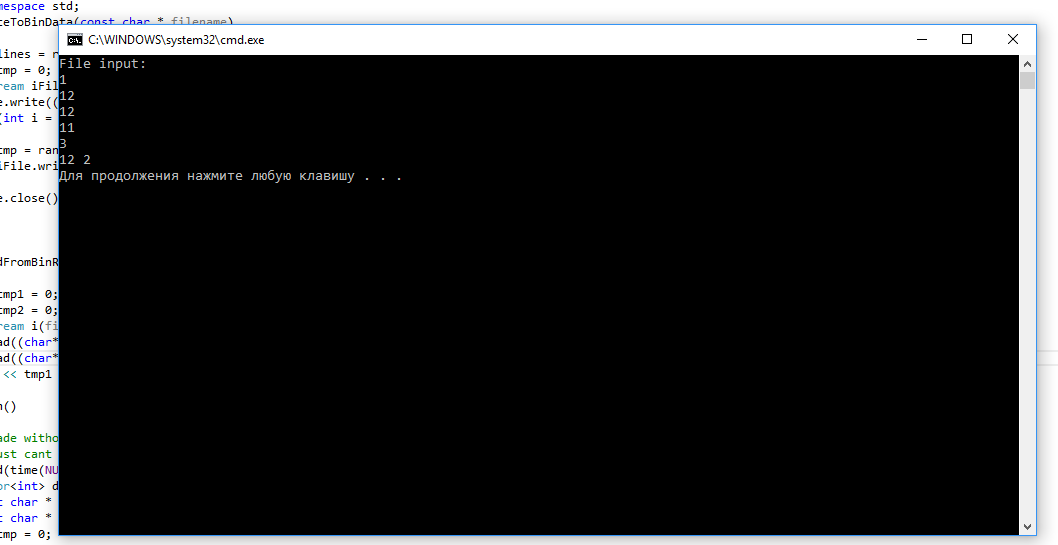
readFromBinResults(oFile);

//result << "max value = " << max << " count = " << count << endl;

//result.close();

//ifstream read("data.txt");

}



**Висновок:** на даній лабораторній роботі я познайомився iз засобами файлового вводу та виводу в мовi програмування С.