**Практична робота № 9 Розроблення програм з використанням файлів**

**Мета:** навчитися складати програми з процедурами і функціями**.**

**Завдання до практичної роботи надається в кінці теоретичної частини.**

**Строк виконання 17.05.2020**

**Перед початком перегляньте лекцію 10.**

**ТЕОРЕТИЧНА ЧАСТИНА**

Розглянемо конкретні приклади використання файлів.

***Приклад 1.*** Записати у файл матрицю matr(2,4) поелементно за рядками, прочитати її з файлу і вивести на екран.

**// *Формування файлу з елементів матриці***

**#include <iostream>**

**#include <fstream>**

**#include <conio.h>**

**#include <Windows.h>**

**using namespace std;**

**int main()**

**{system("color F0");**

**int matr[2][4], i, j;**

**//----------------------------------- запис матриці до файлу**

**ofstream out("filemat.txt", ios::out | ios::binary); // відкриття файлу**

**cout << "Input matricu 2x4;" << endl;**

**for (i = 0; i < 2; i++)**

**{**

**for (j = 0; j < 4; j++)**

**{**

**cin >> matr[i][j]; // введення поточного елементу матриці**

**out << matr[i][j] << " "; // запис до файлу цього елементу**

**}**

**}**

**out.close();**

**//----------------------- виведення матриці на екрані**

**ifstream in("filemat.txt", ios::in | ios::binary); // відкриття файлу**

**for (i = 0; i < 2; i++)**

**for (j = 0; j < 4; j++)**

**in >> matr[i][j]; // читання з файлу елементу матриці**

**in.close();**

**//---------------------- виведення матриці на екрані**

**cout << "\nMatrix matr";**

**for (i = 0; i < 2; i++)**

**{**

**cout << endl;**

**for (j = 0; j<4; j++)**

**cout << matr[i][j] << " ";**

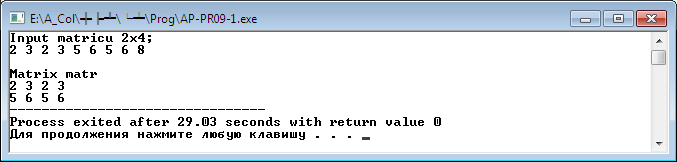
**}**

**getch();**

**return 0;**

**}**

**Результат розв’язання прикладу:**



У цій програмі спочатку елементи матриці з клавіатури вводились у пам’ять комп’ютера, потім кожен з них записувався у файл з ім’ям **filemat**. Для цього попередньо було створено потік **out** і відкрито файл на диску.

Потім було створено потік **in** для зчитування даних з файлу в пам’ять комп’ютера, тобто до матриці **matr[i][j].** Наприкінці програми матрицю виведено на екран.

Зверніть увагу на опис файлів. В лекції 10 ми детально його не розглядали.

Оголошення файлової змінної та відкриття текстового файлу для читання має наступний синтаксис:

**ifstream in(“filemat.txt”, ознака1|ознака2|…|ознакаN);**

У даному записі **in** – назва файлової змінної, **filemat.txt** –назва файлу на диску. Ознаки слугують для визначення прав доступу до файлу. Перелік ознак відкриття файлу та їх значень наведено в табл. 1.

Оголошення файлової змінної та відкриття файлу для запису має синтаксис:

**ofstream f\_out(“filemat.txt”, ознака1|ознака2|…|ознакаN);**

У даному записі **out** – назва файлової змінної, **filemat.txt** –  назва файлу на диску.

Якщо не використовувати ознаки відкриття файлу для читання чи запису, на диску буде створений новий файл із зазначеним ім'ям, або перезаписаний файл із таким же ім'ям, якщо він вже існує.

Наприклад, для того, щоб записати потрібну інформацію в кінець файлу text1.dat, необхідно скористатися фрагментом програмного коду (тут файлова змінна flags):

ofstream flags(“text1.dat”, ios::app);

Таблиця 1 – Ознаки відкриття файлу

| **Ознака** | **Призначення** |
| --- | --- |
| ios::in | Відкриває файл для читання. Вміст файлу зберігається |
| ios::out | Відкриває файл для запису. Якщо файл не існує, то буде створений |
| ios::app | Відкриває файл для дозапису. Дані будуть записані у кінець файлу |
| ios::trunk | Якщо  файл, який відкривають для запису, вже існує, то його вміст буде видалено. |
| ios::nocreate | Забороняє створювати файл, який відкривають |
| ios::inoreplace | Забороняє перезаписувати існуючий  файл |

Файли класифікують за типом компонентів і за методом доступу до них. За типом компонентів розрізняють текстові та бінарні (двійкові) файли, а за методом доступу – файли послідовного і прямого доступу. Ми будемо розглядати файли послідовного доступу. Текстові файли призначені для збереження текстів (наприклад, текстів програм), а бінарні файли використовуються для збереження даних різних типів. Файл бінарний – це лінійна послідовність байтів, що відповідає внутрішньому поданню даних без поділу на рядки. Для завдання бінарного файлу використовується ознака ios:binary (двійковий режим). Якщо задається декілька ознак (як в прикладі), то вони розділяються вертикальною чертою ( | ).

***Приклад 2.*** Записати у файл 5 прізвищ, потім прочитати їх і вивести на екран.

**/\* *програма запису до файлу та читання з файлу масиву з 5 прізвищ* \*/**

**#include <iostream>**

**#include <fstream>**

**#include <conio.h>**

**#include <Windows.h>**

**using namespace std;**

**int main()**

**{system("color F0");**

**char st[5][15];**

**int i;//------------------ запис до файлу**

**ofstream fout("st\_file.dat"); // відкриття файлу**

**if (!fout) cout << "Cannot open file\n";**

**for (i = 0; i <5; i++)**

**{**

**cout << " Enter " << (i+1) << " name\n";**

**cin.getline(st[i],15); // введення поточного прізвища**

**fout << st[i] << '\n'; // запис прізвища до файлу**

**}**

**fout.close();//------------читання файлу та виведення на екран**

**cout << "\nReading file\n\n";**

**ifstream fin("st\_file.dat");**

**if (!fin)cout << "Cannot open file.dat\n";**

**for (i = 0; i < 5; i++)**

**{**

**fin.getline(st[i],15);**

**cout << st[i] << " ";**

**}**

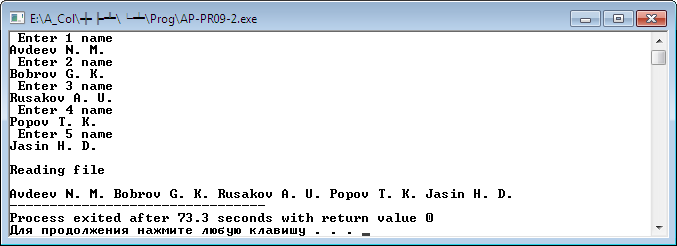
**fin.close();**

**getch();**

**return 0;**

**}**

Результат роботи програми має вигляд:

**.**

У попередніх програмах запис даних у файл та їх читання з файлу здійснювалось послідовно поелементно. Але записати або прочитати декілька даних (наприклад, масив чисел) можна однією операцією. Для цього використовують функції-члени відповідних класів, які мають вигляд:

**in.write((char\*)&p,sizeof(p));** — для запису даних у файл,

**in.read((char\*)&p,sizeof(p));** — для читання даних з файла,

де **іn** — ім’я потоку введення або виведення;

**р** — змінна будь-якого типу, якщо змінна **р** має тип **char[ ],** то операція її приведення не потрібна.

Згадаємо лекцію 9. Коли нам потрібно записати одну символьну змінну ми використовуємо одновимірний масив типу **char[ ]**. Але коли нам потрібно записати масив символьних змінних ми використовуємо двовимірний масив, наприклад,

**char str [5][20] = {“Петренко И. И. “Головко С. С. “, . . . ,};**

де перший індекс визначає номер змінної, а другий максимальну довжину змінної. Функція **getline** працює з типом **char[ ],** саме тому ми так описуємо масив прізвищ.

*Ще одне зауваження* стосується перевірки доступності файлу, це вирази **(!fout)**та**(!fin).** Якщо файл відкрити неможливо, то **fout** та **fin** будуть мати значення false.

**Завдання.**

Модифікуйте свою програму з лабораторної роботи №8 таким чином, щоб окрім виведення на екран дані виводилися до файлу. Виведені дані прочитайте та знов виведіть на екран.

**Увага!** C++Shell не підтримує роботу з файлами.

Для роботи з файлами можна використати безкоштовний компілятор Dev-C++, який потрібно встановити на ваш комп’ютер. Скачати його можна за посиланням:

**https://sourceforge.net/projects/orwelldevcpp/**

Інструкція по роботі з компілятором надається за посиланням (але там забагато реклами іта іншого сміття):

**https://studopedia.ru/18\_64672\_pokrokove-vikonannya-programi.html**

Якщо у Вас нема можливості встановити цю програму, то просто запишіть код виконання операцій роботи з файлами та надішліть на перевірку.