**Практична робота №8. Написання програм з використанням файлів для введення/виведення даних**

**Мета**: навчитися писати програми з використанням з використанням файлів для введення/виведення даних сивів **на мові С++** в **консольному режимі.**

**Завдання.**

1. Прочитати (повторити) основні теоретичні відомості. Переглянути лекцію №10 "Стандартні консольні потоки введення/виведення. Використання файлів для введення/виведення даних".
2. Запустити середовище програмування С++ . Записати програму, що виконує завдання з п.3. В першому рядку кожної програми записати

*// Група № Прізвище*

вказавши номер своєї групи та своє прізвище

1. Написати програму, в яка містить структуру, що описує об'єкт будинок і складається зі змінних, в яких знаходиться інформація про місто (в якому будинок знаходиться), вулиця, номер будинку, кількість квартир, інтернет (проведено чи ні, задати логічною/булевою змінною).
2. Описати файли для виведення та читання даних.
3. Сформувати масив структур (не менше 3), в якому номер будинку, кількість квартир заповнюється випадковими числами з діапазону від 1 до 20 та від 100 до 200, відповідно (див. ПР№6), іншим елементам присвоюються значення в програмі.
4. Відкрити файл для виведення даних та записати до нього інформацію, яка формується у п. b).
5. Закрити файл для виведення даних.
6. Відкрити сформований файл для читання даних та оброблювати прочитані дані, записаны у п. c). Примітка. Дані ви можете читати в нову окрему структуру, якщо ви використовували масив структур в п. а).
7. В програмі запитати кількість об’єктів, про які потрібно вивести інформацію, та вивести перші із запитаної кількості елементи масиву структур з вказівкою індексу масиву, а після нього вся інформація про об’єкт. Значення, що виводяться по об’єкту, розділяються їх табуляцією.
8. Закрити файл для читання даних, а потім знов відкрити і виконати дії за п. h).
9. Після цього підрахувати кількість будинків без інтернету та вивести їх кількість, після цього вивести заголовок "Будівлі з інтернетом" і вивести порядковий номер в списку, місто, вулиця, номер будинку.
10. Закрити файл для читання даних

Для кращого розуміння процесу, що будете програмувати, намалюйте блок-схему.

1. Результати надсилати на електронну адресу викладача [**t.i.lumpova@gmail.com**](mailto:t.i.lumpova@gmail.com)у вигляді cpp-файлу з іменем у форматі

**<Номер групи><Номер лабораторної><Прізвище англійською>**

Наприклад, 31-01Ivanov.cpp.

Іншим рішенням є надсилання поштою посилання на текст програми за URL адреси, яку надає C++Shell, вказавши в темі листа, номер групи прізвище студента та номер ПР.

В темі листа вказати, номер групи, прізвище студента та номер ПР як "ПР№8".

**Строк відсилки ПР для МІВТ/ЕТ-41 17.11.2020**

**МНТ/ЕТ- 41 17.11.2020.**

Всі запитання, що виникнуть, надсилайте на електронну адресу викладача, В темі листа вказати, номер групи, прізвище студента та номер ПР як "ПР№8" В темі листа вказати, номер групи, прізвище студента, номер ПР та фразу "Запитання".

**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 та 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.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

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**

**// відкриття файлу**

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

**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] << " ";**

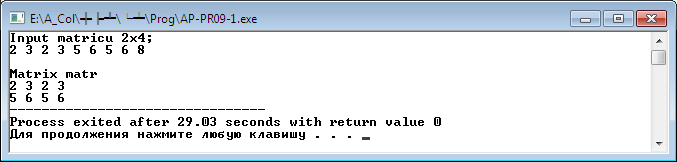
**}**

**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, яка визначає ім’я потоку, та відкриття текстового файлу для читання має наступний синтаксис:

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

У даному записі **in** – назва файлової змінної/ ім’я потоку, **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] << " ";**

**}**

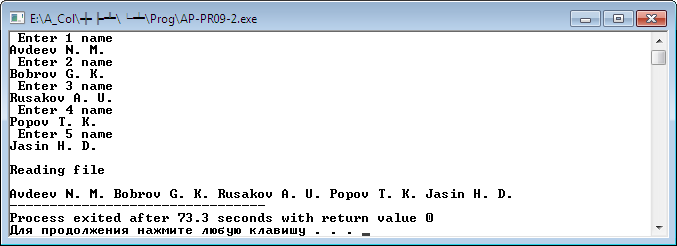
**fin.close();**

**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 файлу,

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**.

***Приклад 3.*** Розробити програму, за допомогою якої здійснюється запис даних типу структура (список прізвищ абонентів та їх телефонів) у файл з ім’ям **struct.txt**.

/\* *програма запису до файлу* struct.txt *даних типу структура — списку прізвищ абонентів та їх номерів телефонів* \*/

**#include <iostream>**

**#include <fstream>**

**#include <string>**

**#include <conio.h>**

**#include <Windows.h>**

**using namespace std;**

**struct telefon**

**{char fio[15]; char tel[10];};**

**main()**

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

**int i;**

**telefon spis[5];**

**ofstream out("struct.txt ");**

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

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

**cout << "Enter " << i+1 << " last name and phone number\n";**

**//------------ Введення прізвища і телефону з клавіатури**

**cin >> spis[i].fio;**

**cin >> spis[i].tel;**

**//------------ Виведення прізвища і телефону до файлу**

**out << spis[i].fio << " " << spis[i].tel << endl;**

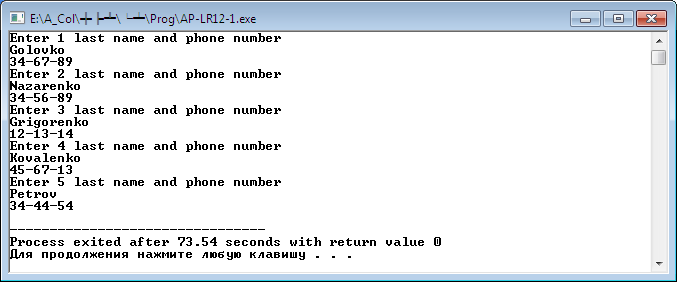
**}**

**out.close();**

**getch ();**

**}**

Результат роботи цієї програми такий:



***Приклад 4.*** Розробити програму читання файлу (**struct.txt**), створеного у прикладі1, і виведення на екран за запитом користувача або списку прізвищ абонентів і їх телефонів, або тільки прізвища і номера телефону потрібного абонента.

/\* *програма читання з файлу даних типу структура — список прізвищ абонентів та номерів телефонів* \*/

**#include <iostream>**

**#include <fstream>**

**#include <string>**

**#include <conio.h>**

**#include <Windows.h>**

**using namespace std;**

**struct telefon**

**{char fio[15]; char tel[10];};**

**main()**

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

**int i, p;**

**char name[15];**

**bool t;**

**telefon spis;**

**//------------------ відкриття раніше створеного файлу**

**ifstream in("struct.txt");**

**if (!in) cout << "\nCannot open file fo reading\n";**

**//------------------ завдання режиму роботи з файлом**

**cout << "Who make: reading list(1) or name(2)\n";**

**cin >> p; //---введення номеру режиму**

**if (p == 1) //---оброблення 1 режиму**

**{**

**//------------------ читання даних з файлу "struct.txt"**

**while (in >> spis.fio >> spis.tel)**

**//------------------ виведення даних на екран**

**cout << spis.fio << " " << spis.tel << endl;**

**}**

**else if (p == 2) //---оброблення 2 режиму**

**{**

**t = true;**

**cout << "Enter name\n";**

**cin >> name; //---введення прізвища**

**//------------------------цикл для читання даних з файлу**

**while (in >> spis.fio >> spis.tel)**

**if (strcmp(spis.fio,name) == 0)**

**{**

**//-----------------------виведення даних на екран**

**cout << spis.fio << " "<<spis.tel<<"\n";**

**t = false;**

**}**

**in.close();**

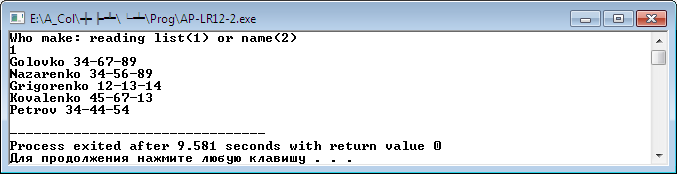
**if (t) cout <<"Name "<< name <<" is not\n";**

**}**

**getch();**

**}**

Результат розв’язання програми для першого режиму її роботи, коли потрібно вивести усі дані з файлу:



Результат розв’язання цієї програми для другого режиму роботи, коли необхідно вивести задане прізвище і телефон, має вигляд:

