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

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

**Строк виконання 28.04.2021**

**Перед початком перегляньте лекцію 10.**

**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.

**Методичні вказівки щодо організації самостійної роботи студентів ЛР№8**

1. Запустити середовище програмування С++ .
2. Записати програму, що виконує 3 завдання з пп.3.1-3.3. В першому рядку кожної програми записати

*// ПТБД-21 ЛР № Прізвище*

вказавши номер своєї групи та своє прізвище.

Як і в ЛР №5, програма повинна запитати номер завдання (число 1, 2, 3 або 0 для закінчення) і в залежності від введеного значення виконувати відповідне завдання. Якщо введений 0 – програма припиняє роботу. Оформити кожне завдання як функцію, яка викликається при обранні відповідної позиції меню.

Вхідні дані ввести, а результати вивести, використовуючи потокове введення-виведення даних.

1. Завдання:
   1. Написати код, який заповнює масив з 10 елементів випадковими числами з діапазону від 200 до 300, виводить всі елементи масиву з вказівкою індексу масиву, розділивши їх табуляцією, виконує пошук мінімального та максимального числа та їх виведення.

Вивести масив до файлу, а потім прочитати файл та вивести дані файлу на консоль.

* 1. Написати код, який заповнює масив з 10 елементів випадковими числами з діапазону від -20 до 20, виводить всі елементи масиву з вказівкою індексу масиву, розділивши їх табуляцією, виконує заміну всіх негативних чисел на 55, а позитивні робить негативними. Вивести отриманий масив аналогічно заповненому.

Вивести масив до файлу, а потім прочитати файл та вивести дані файлу на консоль.

* 1. Написати код, який заповнює по 2 змінні (можна ввести з консолі або проініціалізувати) типу структура з інформацією про особу. Структура повинна містити поля:
* код (порядковий номер),
* прізвище та ініціали,
* стать (1 – жіноча, 2 чоловіча),
* телефон,
* адреса,
* професія.

Вивести всі значення по особах.

Вивести масив до файлу, а потім прочитати файл та вивести дані файлу на консоль.

**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 <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**

**// відкриття файлу**

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

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

**{**

**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] << " ";**

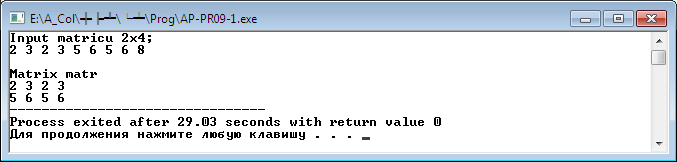
**}**

**}**

**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::noreplace | 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 рискою ( | ).

***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";else**

**{**

**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";**

**else**

**{**

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

**{**

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

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

**}**

**fin.close();**

**}**

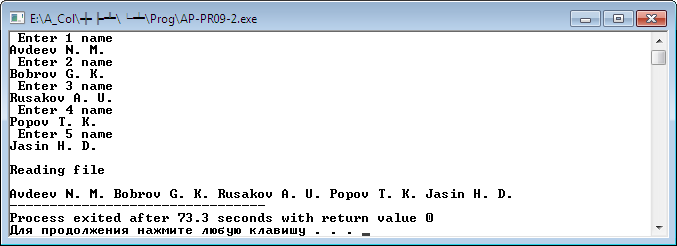
**}**

**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. Записати у файл три рядки тексту.**

**#include <iostream>**

**#include <fstream>**

**#include <conio.h>**

**using namespace std;**

**int main()**

**{**

**ofstream fout("output.txt");**

**fout<<"Prog na C++ "<<"I like"<<endl;**

**fout<<"My name is"<<endl;**

**fout<<"I am a student"<<endl;**

**fout.close();**

**return 0;**

**}**

Для того щоб прочитати створений файл, знадобитися виконати аналогічні кроки, що й при запису у файл з невеликими змінами:

ifstream fin("cppstudio.txt");

fin >> buff;

fin.close().

1. створити об'єкт класу ifstream і пов'язати його з файлом, який відкриємо в режимі для читання:
2. прочитати файл;
3. закрити файл.

**Приклад 4. Прочитати файл, записаний в результаті виконання прикладу 3.**

**#include <iostream>**

**#include <fstream>**

**#include <conio.h>**

**#include <Windows.h>**

**using namespace std;**

**int main()**

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

**ifstream fin("output.txt");**

**char line[64];**

**while (!fin.eof())**

**{**

**fin.getline(line,sizeof(line));**

**cout<<line<<endl;**

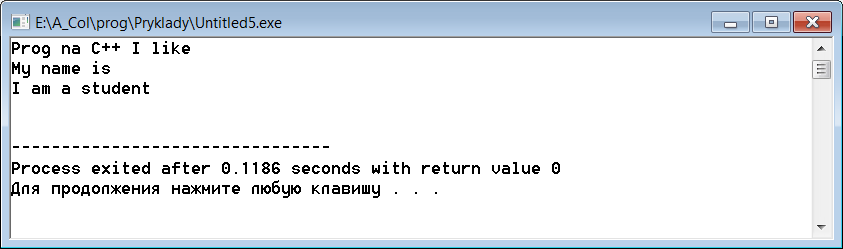
**}**

**fin.close();**

**return 0;**

**}**

У даному випадку зчитування інформації з файлу відбувається доти, доки не буде досягнуто кінця файлу.



Для використання функцій введення/виведення необхідно підключити відповідний заголовочний файл:

#include <cstdio> //або <stdio.h>

Наявні функції бібліотеки дозволяють працювати й з стандартними вхідним та вихідним потоками: stdin – клавіатура, stdout – екран, stderr – помилки.

Бібліотечні функції, оголошені в conio.h, різняться в залежності від компілятора.

*Наприклад:*

* getchar(), putchar() – читання та запис символу;
* gets(), puts() – читання та запис рядка;
* scanf(), printf() – форматоване введення та виведення даних
* Обробка інформації, що зберігається у вигляді файлу передбачає наступні дії:
* визначення змінної – файлового покажчика;
* відкривання та закривання потоку;
* введення-виведення (символів, рядків, форматованих даних, порцій даних певної довжини);
* аналіз можливих помилок операцій введення-виведення;
* керування буферізацією потоку (розміром буферу);
* керування буферним покажчиком.

Мова С++ підтримує дві системи введення/виведення. Перша – спадок С. Друга – власна, об`єктно-орієнтована. Дозволяється поєднувати в одній програмі.

До переваг об`єктно-орієнтованої системи введення/виведення можна віднести: простоту використання у простих випадках, можливості перевизначення для власних класів. Але вона вимагає розуміння об`єктів та класів.

Введення/виведення в “*стилі С*” зручне при форматованому обміну, дозволяє побачити й зрозуміти роботу з файлами, потрібне для розуміння та підтримки накопиченого програмного забезпечення.

Потоки бувають двох типів: текстові та двійкові.

Кожний потік має керівну структуру типу FILE , що містить усю необхідну інформацію для роботи з ним. Змінна, що буде представляти потік визначається як покажчик на структуру типу FILE.

Наприклад:  
 FILE \*fp;

Змінна fp зображує потік у подальшій роботі з файлом.

Опис типу FILE , а також прототипи більшості функцій, макросів та констант файлової системи містяться у заголовному файлі <cstdio> (та <stdio.h>).

***Контрольні запитання для самоперевірки***.

1. Які види потоків ви знаєте?
2. Для чого призначені стандартні потоки?
3. Який файл необхідно включати у програму для використання стандартних потоків?
4. Назвіть класи вхідних та вихідних потоків.
5. Який клас є базовим для потоків?
6. Для чого призначені маніпулятори потоків, і який файл необхідно включати у програму для їх використання?
7. Для чого призначені маніпулятори endl, setprecision, setw?
8. Які є файли за способом доступу?
9. Назвіть класи для створення файлових потоків і їх призначення.
10. Які дії необхідно виконати для використання файлів у програмі?
11. Які існують режими відкриття файлів?
12. Для чого призначений метод open() і які він має параметри?
13. Які способи відкриття та закриття файлу ви знаєте?