**Лекція № 11. Бінарні та текстові файли.**

Файл – це впорядкована сукупність даних, що зберігається на диску і займає іменовану область зовнішньої пам'яті. Згідно з термінологією, прийнятою в програмуванні, файл – це іменований блок інформації, який зберігається на носії інформації. Файл обов'язково має ім'я і може мати будь-який розмір (максимальна довжина імені та розміру файлу обмежується властивостями конкретної файлової системи). Файли можуть містить текстову, числову, графічну інформацію.

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. Текстові файли призначені для збереження текстів (наприклад, текстів програм), а бінарні файли використовуються для збереження даних різних типів.

Файл текстовий є сукупністю символьних рядків змінної довжини. Кожен рядок завершується маркером кінця рядка – спеціальною парою керуючих символів: #13 (повернення каретки) та #10 (переведення рядка). Наприкінці файлу записується маркер кінця файлу – керуючий символ #26.

Файл бінарний – це лінійна послідовність байтів, що відповідає внутрішньому поданню даних без поділу на рядки.

**Функції обробки файлів**

У мові С++ використовуються функції роботи з файлами, запозичені з мови програмування C. Загальновживані функції роботи з файлами наведено в табл. 1.

Таблиця 1 – Основні функції роботи з файлами

| **Функція** | **Призначення** |
| --- | --- |
| fopen() | Відкриває потік |
| fclose() | Закриває потік |
| fputc() | Записує символ у потік |
| fgetc() | Зчитує символ з потоку |
| fwrite() | Записує блок даних у потік |
| fread() | Зчитує блок даних з потоку |
| fseek() | Встановлює індикатор позиції файлу на заданий байт у потоці |
| fprintf() | Робить для потоку те, що функція printf() робить для консолі |
| fscanf() | Робить для потоку те, що функція scanf() робить для консолі () |
| feof() | Повертає значення true, якщо досягнуто кінець файлу |
| ferror() | Повертає значення true, якщо виникла помилка |
| rewind() | Встановлює індикатор позиції файлу у початок файлу |
| remove() | Видаляє файл |

**Функції, призначені для відкриття / закриття файлів**

Функція fopen() виконує наступні дії:

* відкриває потік;
* пов'язує файл з потоком;
* повертає покажчик типу FILE на цей потік.

Прототип функції fopen():

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

У цьому записі параметр **filename** вказує на ім'я файлу, що відкривається, параметр **mode** – на рядок, що містить потрібний режим відкриття файлу. Можливі значення режиму відкриття файлу наведені у табл. 2. Параметр **filename** повинен представляти рядок символів, який визначає ім'я файлу, допустиме у даній операційній системі. Цей рядок може містити специфікацію шляху до файлу, якщо діюче програмне середовище підтримує таку можливість.

Таблиця 2– Режими роботи з файлами

| **mode** | **Призначення** |
| --- | --- |
| "r" | Відкриває текстовий файл для зчитування |
| "w" | Створює текстовий файл для запису |
| "а" | Відкриває текстовий файл для запису у кінець файлу |
| "rb" | Відкриває двійковий файл для зчитування |
| "wb" | Створює двійковий файл для запису |
| "ab" | Відкриває двійковий файл для запису у кінець файлу |
| "r+" | Відкриває текстовий файл для зчитування і запису |
| "w+" | Створює текстовий файл для зчитування і запису |
| " a+ " | Відкриває текстовий файл для зчитування і запису у кінець файлу |
| "r+b" | Відкриває двійковий файл для зчитування і запису |
| "w+b" | Створює двійковий файл для зчитування і запису |
| "a + b" | Відкриває двійковий файл для зчитування і запису у кінець файлу |

У випадку успішного відкриття вказаного файлу функція **fopen**() повертає покажчик FILE. Цей покажчик ідентифікує файл. Якщо файл не вдається  відкрити, наприклад, якщо файл із заданим іменем відсутній на диску, повертається нульовий покажчик.

**Приклад 1. Відкриття текстового файлу test для запису:**

**fp = fopen("test", "w");**

Функція fopen() має режими роботи як з текстовими, так і з двійковими (бінарними) файлами.

Закриття файлу здійснюється функцією **fclose(),** що має наступний прототип:

**іnt fclose (FILE \* Stream);**

У якості параметра функції виступає покажчик на файл, що закривається. Функція повертає 0 якщо файл був успішно закрито, і  значення 1 – в іншому випадку.

Приклад виклику функції закриття файлу:

fclose(p1);

Під час виклику функції  fclose()  звільняється  блок  керування файлом, що  пов'язаний  з  потоком,  що  робить  його  доступним  для  повторного використання.

**Введення/виведення даних**

Функція **fputc()** використовують для виведення символів у потік, який заздалегідь відкрито для запису за допомогою функції fopen().

Прототип функції fputc():

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

У цьому  записі параметр  fp представляє собою файловий покажчик, що повертається функцією fopen(), а параметр  ch –  символ, що виводиться. Файловий покажчик повідомляє функції fputc(), в який дисковий файл необхідно записати символ.. При  успішному  виконанні  операції  виведення  функція  fputc()  повертає записаний у файл символ, інакше – значення EOF.

Функція  **fgetc()**  використовують  для  зчитування  символів  з потоку,  відкритого в режимі  зчитування  за допомогою функції  fopen().

Прототип функції fgetc():

**int fgetc(FILE \*fp);**

У цьому  записі параметр  fp означає файловий покажчик, що повертається fopen(). Під  час  виникнення  помилки  або  досягнення кінця файлу функція fgetc() повертає значення EOF. Для того, щоб зчитати весь вміст текстового файлу можна використати фрагмент програмного коду:

**ch = fgetc(fp);**

**while(ch != EOF)**

**{**

**ch= fgetc(fp);**

**}**

Функція **fprintf()** форматує і друкує у вихідний потік набори символів і значень. Кожен аргумент (якщо він є) перетворюється і виводиться згідно заданої специфікації формату.

Прототип функції fprintf():

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

Функція **fscanf()** зчитує дані з поточної позиції потоку в місце, обумовлене значенням аргументів (якщо вони є). Кожен аргумент повинен бути покажчиком на змінну і тип, який відповідають типу, заданому в рядку формату.

Прототип функції fscanf():

**int fscanf (stream, format-string [, argument ...]);**

Таким чином призначення функцій fprintf() та fscanf() аналогічно функціям printf()  і scanf(), за винятком  того, що вони працюють з файлами.

**Приклад 2. Дано два текстових файли, що містять цілі числа, впорядковані за зростанням. Необхідно об'єднати ці файли в третій, що будуть містити впорядковані за зростанням числа з перших двох файлів.**

**#include <iostream>**

**#include <fstream>**

**#include <string>**

**#include <Windows.h>**

**using namespace std;**

**int main()**

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

**FILE \*f1,\*f2,\*r;**

**int a1,a2;**

**f1= fopen("file1.txt","r");**

**if(f1==NULL) return 1;**

**f2 = fopen("file2.txt","r");**

**if(f2==NULL)**

**{**

**fclose(f1);**

**return 2;**

**}**

**r = fopen("result.txt","w");**

**if(r==NULL){fclose(f1);fclose(f2);return 3;}**

**fscanf(f1,"%d",&a1);**

**fscanf(f2,"%d",&a2);**

**while(1)**

**{**

**if(a1<a2)**

**{**

**fprintf(r,"%d ",a1);**

**if(fscanf(f1,"%d",&a1)==EOF) break;**

**}**

**else**

**{**

**fprintf(r,"%d ",a2);**

**if(fscanf(f2,"%d",&a2)==EOF) break;**

**}**

**}**

**if(feof(f1))**

**{**

**while(!feof(f2))**

**{**

**fprintf(r,"%d ",a2);**

**fscanf(f2,"%d",&a2);**

**}**

**fprintf(r,"%d ",a2);**

**}**

**else**

**if(feof(f2))**

**{**

**while(!feof(f1))**

**{**

**fprintf(r,"%d ",a1);**

**fscanf(f1,"%d",&a1);**

**}**

**fprintf(r,"%d ",a1);**

**}**

**fclose(f1);**

**fclose(f2);**

**fclose(r);**

**return 0;**

**}**

Функція  **ferror()**  використовують  для  визначення факту  виникнення  помилки у процесі виконання операції з файлом.

Її прототип має такий вигляд:

**int ferror(FILE \*fp);**

У цьому записі параметр  fp дійсний файловий покажчик. Функція  ferror() повертає значення  true, якщо у процесі виконання останньої файлової операції відбулася  помилка;  інакше  значення  false.  Оскільки  виникнення  помилки можливе  у  процесі  виконання  будь-якої  операції  з файлом, функцію  ferror() необхідно викликати відразу після кожної функції оброблення файлів; інакше інформацію про помилку можна просто втратити.

Функція  **rewind()**  переміщує  індикатор  позиції  файлу  у  початок  файлу, що задається як аргумент.

 Її прототип має такий вигляд:

**void rewind(FILE \*fp);**

У цьому записі параметр fp представляє собою дійсний файловий покажчик.

Функція **fseek()** призначена для встановлення індикатора позиції файлу у необхідному місці. Таким чином, система введення-виведення у мові С++ подібно до мови С дає змогу виконувати операції зчитування і запису даних з довільним доступом.

Прототип функції fseek():

**int fseek(FILE \*fp, long numbytes, int origin);**

У цьому  записі параметр  fp означає файловий покажчик, що повертається функцією fopen(), параметр numbytes визначає кількість байтів щодо початкового положення, які задаються параметром origin. Параметр origin може приймати одне наступних макроімен (визначених у заголовку stdio.h):

* SEEK\_SET – константа 0 – пошук з початку файлу;
* SEEK\_CUR– константа 1 – пошук з поточної позиції;
* SEEK\_END– константа 2 – пошук з кінця файлу.

Нульове  значення  результату  функції  свідчить  про  успішне  виконання функції  fseek(),  а  ненульове  –  про  виникнення  збою.  Як  правило,  функцію fseek() не рекомендується використовувати для файлів, відкритих у текстовому режимі, оскільки перетворення символів може призвести до помилкових переміщень індикатора позиції у файлі. Тому краще використовувати цю функцію для файлів, відкритих у двійковому (бінарному) режимі.

**Приклад 3. Дано бінарний файл, що містить дійсні числа. Поміняти в ньому місцями максимальний і мінімальний елементи.**

**#include "stdafx.h"**

**void main()**

**{**

**FILE \*f;**

**int i=0, pmin=0, pmax=0;**

**double max,min,v;**

**f = fopen("file.txt","r+b");**

**if(f==NULL) return;**

**fread(&v,sizeof(double),1,f);**

**max = v;**

**min = v;**

**i=1;**

**while(!feof(f))**

**{**

**fread(&v,sizeof(double),1,f);**

**if(max<v) pmax = i;**

**if(min>v) pmin = i;**

**i++;**

**}**

**fseek(f,pmax\*sizeof(double),0);**

**fwrite(&min,sizeof(double),1,f);**

**fseek(f,pmin\*sizeof(double),0);**

**fwrite(&max,sizeof(double),1,f);**

**fclose(f);**

**return;**

**}**

**Введення / виведення бінарних файлів стандартними засобами**

У файловій системі мови С передбачено дві функції  fread()  і  fwrite(), які дають  змогу  зчитувати  і  записувати  блоки  даних. Ці  функції  подібні  С++функціям read() і write(). Їх прототипи мають такий вигляд:

**size\_t fread(void \*buffer, size\_t num\_bytes, size\_t count, FILE \*fp);**

**size\_t fwrite(const void \*buffer, size\_t num\_bytes, size\_t count, FILE \*fp);**

Під  час  виклику  функції  **fread()**  параметр  buffer  є  покажчиком  на  область пам'яті, яка призначена для прийому даних, які зчитуються з файлу. Функція fread() повертає кількість зчитаних об'єктів, яка може опинитися  менше  заданого  значення  count,  якщо  у  процесі  виконання  цієї операції виникла помилка або був досягнуто кінець файлу.

Під час виклику функції  **fwrite()** параметр buffer є покажчиком на  інформацію, яка підлягає запису у файл. Ця функція записує count об'єктів завдовжки num\_bytes у потік, яка адресується файловим покажчиком  fp. Функція  fwrite() повертає кількість записаних об'єктів, яка буде дорівнює значенню count, якщо у процесі виконання цієї операції не було помилки.

Якщо під час виклику функцій fread() і fwrite() файл був відкрито для виконання  двійкової  операції,  то  вони  можуть  зчитувати  або  записувати  дані будь-якого  типу.

**Приклад 4. Запис значення типу float у двійковий (бінарний) файл.**

**#include "stdafx.h"**

**int main()**

**{**

**FILE \*fp;**

**float f = 12.23F;**

**if((fp=fopen("test","wb"))==NULL)**

**{**

**printf("Не вдається відкрити файл" << endl;);**

**return 1;**

**}**

**fwrite(&f,sizeof(float), 1, fp);**

**fclose(fp);**

**sleep(2000);**

**return 0;**

**}**

**Приклад 5. Запис масиву у бінарний файл.**

**#include "stdafx.h"**

**int main()**

**{**

**register int i;**

**FILE \*fp;**

**float balance[100];**

**//Відкриваємо файл для запису.**

**if((fp=fopen("balance","w"))==NULL)**

**{**

**printf("Не вдається відкрити файл" << endl;);**

**return 1;**

**}**

**for(int i=0; i<100; i++) balance[i] = (float) i;**

**//зберігаємо весь масив balance.**

**fwrite(balance, sizeof balance, 1, fp);**

**fclose(fp);**

**//Обнуляємо масив.**

**for(int i=0; i<100; i++) balance[i] = 0.0;**

**//Відкриваємо файл для зчитування.**

**if((fp=fopen("balance","r"))==NULL)**

**{**

**printf("Не вдається відкрити файл" << endl;);**

**return 1;**

**}**

**//зчитуємо весь масив balance.**

**fread(balance, sizeof balance, 1, fp);**

**// Відображаємо вміст масиву.**

**for(int i=0; i<100; i++) printf("%f ", balance[i]);**

**fclose(fp);**

**getch();**

**return 0;**

**}**

**Приклад 6. Читання/запису масиву структур у файл. Функції write(), read()**

У прикладі використовуються функції write(), read() для роботи зі структурою типу BOOK, а саме:

* запис масиву типу BOOK у файл, що складається з трьох структурних змінних;
* читання масиву структур типу BOOK з файлу.

**#include <iostream>**

**#include <fstream>**

**#include <string>**

**using namespace std;**

**// Структура BOOK**

**struct BOOK**

**{**

**char title[100]; // назва книги**

**char author[70]; // автор**

**int year; // рік випуску**

**float price; // вартість книги**

**};**

**// Запис масиву структур в файл з допомогою функції write()**

**// читання масиву структур з файлу з допомогою функції read()**

**bool Example7(const char \* filename)**

**{**

**// продемонстровано запис в файл та читання з цього файлу масиву структур типу BOOK**

**// створити масив 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; // кількість елементів у масиві B**

**BOOK C[3]; // інший масив, з якого буде виконуватись читання**

**int n2; // кількість елементів у масиві C**

**int i;**

**// 1. Запис масив структур в файл**

**// outF - екземпляр файлу, в який здійснюється запис**

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

**if (!outF) return false;**

**// записати значення n**

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

**// запис масиву B[] в файл wf**

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

**{**

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

**}**

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

**// після закінчення роботи з файлом його потрібно закрити (обов'язково)**

**outF.close();**

**// 2. Читання масиву структур з файлу**

**// inF - екземпляр файлу, з якого здійснюється читання**

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

**if (!inF) return false;**

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

**// Спочатку прочитати кількість записаних структур**

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

**// цикл читання масиву структур в змінну C**

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

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

**inF.close(); // закрити файл**

**// вивід масиву C на екран**

**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");**

**}**

**Приклад 7. Запису/читання масиву чисел типу double**

Демонструється:

* запис у файл масиву **M** чисел типу double функцією write();
* читання з файлу масиву чисел типу double функцією read().

Файл відкривається у двійковому форматі.

**#include <iostream>>**

**#include <fstream>**

**using namespace std;**

**// використання функцій read(), write() для запису/читання масиву чисел**

**bool Example8(const char \* filename)**

**{**

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

**int i;**

**int n = 5; // кількість елементів у масиві M**

**// 1. Запис масиву в файл**

**// 1.1. Створити екземпляр outF, зв'язаний з файлом filename**

**ofstream outF(filename, ios::out | ios::binary); // для запису, двійковий формат**

**// 1.2. Перевірка, чи відкривається файл**

**if (!outF)**

**{**

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

**return false;**

**}**

**// 1.3. Записати кількість елементів у масиві M**

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

**// 1.4. Записати увесь масив в файл**

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

**outF.close(); // закрити файл**

**// 2. Читання даних з файлу filename в масив M2**

**double M2[5];**

**int n2;**

**// 2.1. Відкрити файл для читання**

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

**// 2.2. Перевірка чи файл відкрито**

**if (!inF)**

**{**

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

**return false;**

**}**

**// 2.3. Зчитати к-сть елементів у масиві**

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

**// 2.4. Зчитати дані з файлу в масив M2**

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

**inF.close(); // закрити файл**

**// 2.5. Вивести масив M2 на екран**

**cout << "Array M2:\n";**

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

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

**cout << endl;**

**return true;**

**}**

**void main()**

**{**

**Example8("file8.bin");**

**}**

Результат роботи програми:

Array M2:

2.44 3.85 -3.23 11.85 3.38

***Контрольні запитання для самоперевірки***.

1. Як класифікують файли за типом?
2. Як працює функція fopen()?
3. Чим різняться текстові та бінарні файли?
4. Яке значення повертає функція fgetc() під  час  виникнення  помилки  або  досягнення кінця файлу?
5. Яка функція призначена для встановлення індикатора позиції файлу у необхідному місці?
6. Яку функцію використовують  для  визначення факту  виникнення  помилки у процесі виконання операції з файлом?

**Для самостійного вивчення** *(2 години)*: Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

**Рекомендована література**

1. Ковалюк Т. В. Алгоритмізація та програмування: Підручник. — Львів: «Магнолія 2006», 2013. — 400 с., ил.
2. Павловская Т.А. С/С++. Программирование на языке высокого уровня. СПб.: Питер, 2003. – 461 с. URL: <http://www.ph4s.ru/bookprogramir_1.html>
3. Вступ до програмування мовою С++. Організація обчислень: навч. посіб. / Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставовський. – К. : Видавничо-поліграфічний центр "Київський університет", 2012. – 175 с. с.: іл. ISBN (укр.) . URL: <http://csc.knu.ua/uk/library/books/belov-24.pdf>
4. Джейс Либерти Освой самостоятельно С++ за 21 день: 3-е изд. пер. с англ.: Уч. пос. – М.: Издательский дом „Вильямс”, 2001. – 816 с.: ил..

5. Дейтел Х., Дейтел П. Основы программирования на С++. – М.: Бином, 1999. – 1024 с. URL: <http://ijevanlib.ysu.am/wp-content/uploads/2018/03/deytel.pdf>