



ОПРАЦЮВАННЯ ФАЙЛІВ ЗАСОБАМИ C++

```
#include <fstream>
```

```
ifstream f;
```

або так:

```
ofstream f;
```

або так:

```
fstream f;
```

функції опрацювання потоку викликаються як окрема команда

`f.<функція>(<аргументи>);`

- ім'я змінної, пов'язаної із файловим потоком (файлової змінної);
- ім'я відповідної функції та її аргументи.

Потік – це набір символів, які передаються від деякого довільного пристрою до іншого пристрою.

Терміни файловий потоковий тип, потоковий тип, тип файлового потоку, тип потоку, файловий потоковий клас, клас файлового потоку, потоковий клас, клас потоку – еквівалентні.

Файлова змінна == змінна файлового потоку == файловий потік.

`#include <fstream>`

`fstream` (file stream – файловий потік) – дозволяє як зчитувати, так і записувати дані (файловий потік введення-виведення)

`ifstream` (input file stream – потік файлового введення) – дозволяє лише зчитувати дані з файлу.

`ofstream` (output file stream – потік файлового виведення) – дозволяє лише записувати дані у файл

Файлова змінна, що позначатиме файл, який можна і писати і читати,

`fstream f;`

Файлова змінна, що позначатиме файл, який можна лише читати

`ifstream fin;`

Файлова змінна, що позначатиме файл, який можна лише писати

`ofstream fout;`

Опрацювання файлу



Відкриття файлу – виконується за допомогою функції `open()`

Файлова_змінна.`open("ім'я_файлу"[, режим_відкриття]);`

```
f.open("1.txt");    →    fstream f.open("1.txt");  
fin.open("2.txt");    . . .  
fout.open("3.txt");    if (!f.is_open())  
                        {  
                        cout << "Файл не відкрито!" << endl;  
                        return;  
                        }
```

Закриття файлу – виконує функція `close()`

Файлова_змінна.`close();`

Наприклад `f.close();`

Коли файл закривається, всі дані, які програма записувала у цей файл, скидаються на диск, і операційна система оновлює запис в каталозі для цього файлу. *Якщо файлова змінна – локальна, тобто оголошена у деякому блоці*, то при виході з блоку така змінна буде знищена (як і кожна локальна змінна). При цьому автоматично закривається файл, пов'язаний із цією змінною (лише всередині блоку).

Режими відкриття файлу



Режим відкриття	Призначення
<code>ios::app</code>	Відкриває файл в режимі додавання, файловий вказівник розміщується в кінці файлу.
<code>ios::ate</code>	Розміщує файловий вказівник в кінці файлу.
<code>ios::binary</code>	Відкрити файл як бінарний.
<code>ios::in</code>	Відкриває файл для введення (зчитування).
<code>ios::nocreate</code>	Якщо вказаний файл не існує – то не створювати нового файлу і повернути помилку.
<code>ios::noreplace</code>	Якщо вказаний файл існує – то не замінювати його, операція відкриття файлу має бути перервана і має повернути помилку.
<code>ios::out</code>	Відкрити файл для виведення (запису).
<code>ios::trunc</code>	Перезаписати вміст існуючого файлу.

Вказані константи можна поєднувати

```
fstream f(const char* name, ios::openmode mode = ios::in | ios::out);
```


За умовчанням, з файловими потоковими типами пов'язані певні режими:

```
fstream f(const char* name, ios::openmode mode = ios::in | ios::out);  
ifstream f1(const char* name, ios::openmode mode = ios::in);  
ofstream f2(const char* name, ios::openmode mode = ios::out | ios::trunc);
```

- потоковий тип **fstream** дозволяє одночасно читати і писати файл (режим `ios::in | ios::out`);
- потоковий тип **ifstream** дозволяє лише читати файл (режим `ios::in`), при цьому файл має існувати;
- потоковий тип **ofstream** дозволяє лише писати файл (режим `ios::out | ios::trunc`), при цьому, якщо файл не існував, то він буде створений, а якщо існував – то його вміст буде очищено (`ios::trunc`).

Якщо файл потрібно відкрити в режимі до-запису у кінець файлу:

```
fstream f("1.txt", ios::out | ios::app);  
ofstream f("1.txt", ios::app);
```

Файли, які відкриваються для виведення (`ios::out`), створюються, якщо вони ще не існують.

Типовою файловою операцією є *зчитування вмісту файлу, поки не буде досягнуто кінець файлу*. Для визначення кінця файлу можна використовувати функцію **eof()** файлового потоку. *повертає значення 0, якщо ще не досягнуто кінця файлу, та 1, якщо досягнуто кінець файлу*.

```
while (!f.eof()) // поки не кінець файлу
{
    ... // опрацювання файлу
}
```

При використанні потоків в мові C++ стан «кінець файлу» визначається за поточною позицією файлового вказівника зчитування. Наступний фрагмент коду містить умову, що дає змогу коректно вивести вміст файлу за умови наявності пробілу після останнього символу файлу

```
int k;
while (!f.eof())
{
    f >> k;
    if (!f) // якщо зчитування не вдалося, умова ! f стає істинною
        break; //вихід з циклу
    cout << k << " ";
}
```

Передавання файлів у функцію

1) передати файлову змінну, пов'язану з файлом.

```
void fprocess(ifstream& fin)
{
    // опрацювання файлу
}
int main()
{
    ifstream fin("1.txt");
    fprocess(fin);
    fin.close();
    return 0;
}
```

2) передати оголошену файлову змінну, ще не пов'язану з файлом.

```
void fprocess(ifstream& fin)
{
    fin.open("1.txt");
    // опрацювання файлу
}
int main()
{
    ifstream fin;
    fprocess(fin);
    return 0;
}
```

3) передати літерний рядок – ім'я файлу.

```
void fprocess(const char* filename)
{
    ifstream fin(filename);
    // опрацювання файлу
}
int main()
{
    char filename[] = "1.txt";
    fprocess(filename);
    return 0;
}
```

Параметр – файловий потік обов'язково слід передавати як посилання:

Робота з тестовими файлами

`ifstream fin("1.txt");` – відкриття файлового потоку введення (відкриття файлу для зчитування)

`ofstream fout("1.txt");` – відкриття файлового потоку виведення (відкриття файлу для запису):

`ofstream f("1.txt", ios::app);` – відкриття файлового потоку виведення в режимі дозапису у кінець файлу

`fstream fin("1.txt");` – відкриття файлового потоку введення-виведення

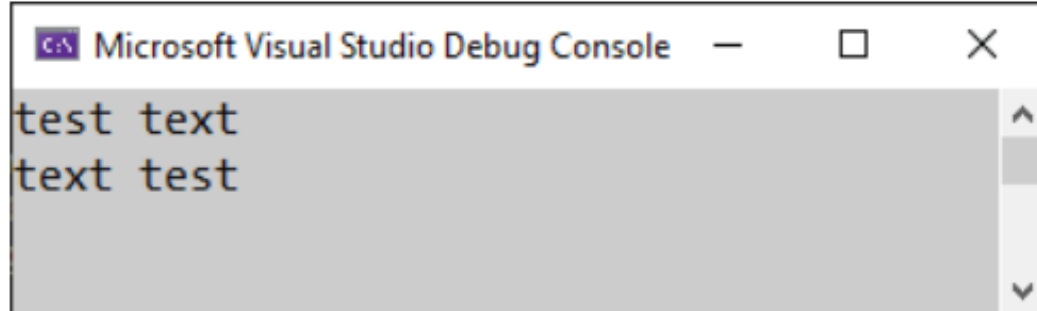
- *при введенні (зчитуванні)* даних з файлу кожна пара символів '\r' + '\n' (повернення каретки + переведення рядка) перетворюється у символ переведення рядка ('\n');

- *при виведенні (запису)* даних у файл кожний символ переведення рядка ('\n') перетворюється у пару символів '\r' + '\n' (повернення каретки + переведення рядка).

Всі команди потокового введення з клавіатури можна застосовувати і для потокового зчитування з файлу.

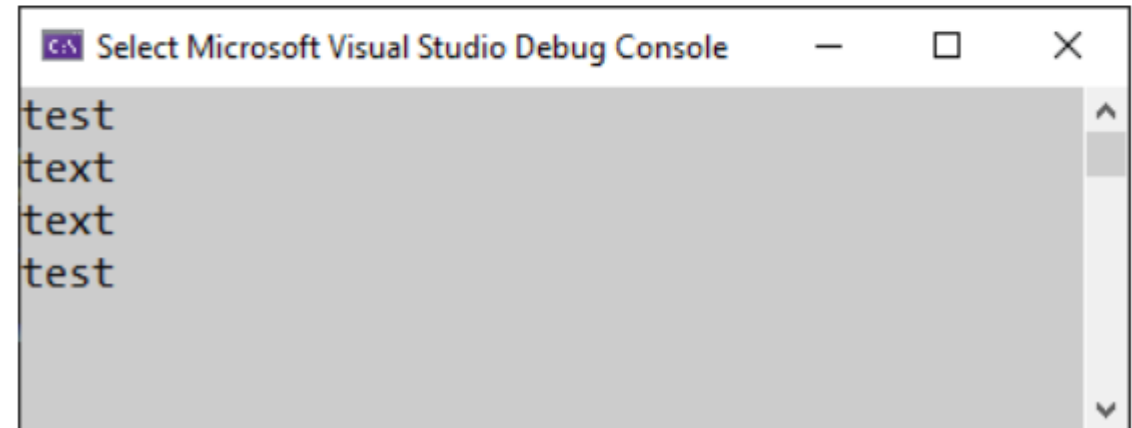
Приклад зчитування даних з текстового файлу

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    ifstream fin("1.txt");
    char line[80];
    while (fin.getline(line, sizeof(line)))
    {
        cout << line << endl;
    }
    return 0;
}
```



зчитує вміст текстового файлу 1.txt по одному слову:

```
#include <iostream>
#include <fstream>
using namespace std;
int main()
{
    ifstream fin("1.txt");
    char word[64];
    while (!fin.eof())
    {
        fin >> word;
        cout << word << endl;
    }
    return 0;
}
```



Робота з бінарними файлами

Для відкриття бінарного файлового потоку слід вказати режим `ios::binary`,

```
fstream f("test.dat", ios::binary);
```

Загальний вигляд команди *запису даних у бінарний файл*:

```
Файлова_змінна.write(адреса_області_пам'яті, sizeof(область_пам'яті));
```

де:

- `Файлова_змінна` – файлова змінна, пов'язана із відкритим бінарним файлом;
- `адреса_області_пам'яті` – вказівник на змінну (адреса змінної), значення якої буде записане у файл;
- `sizeof(область_пам'яті)` – розмір змінної – тої області пам'яті, вміст якої буде записаний у файл.

Якщо в процесі запису нові дані перекрили мітку «кінець файлу», то розмір файлу автоматично збільшується

```
#include <string>
#include <fstream>

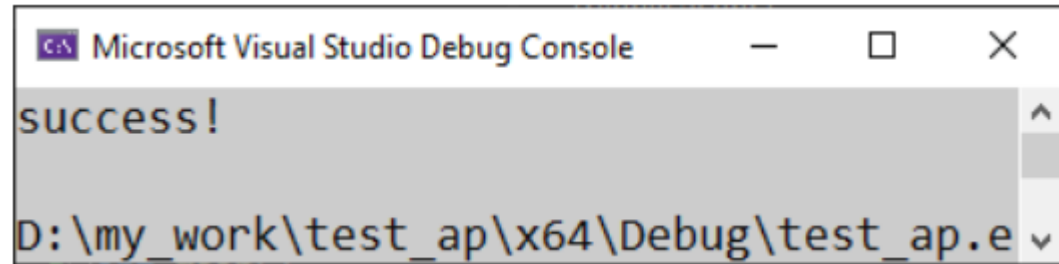
using namespace std;

struct Employee
{
    string name;
    int age;
    float salary;
};

int main()
{
    Employee worker = { "Petrenko", 23, 12000.00 };
    // Відкриття файлу для запису в режимі бінарного запису
    fstream f("test.dat", ios::out | ios::binary);

    // Перевірка, чи файл відкрито
    if (!f.is_open()) {
        cout << "failed to open" << endl;
        return 1;
    }
    // Запис даних у файл
    f.write((char*)&worker, sizeof(Employee));
    // Закриваємо файл
    f.close();
    cout << "success!" << endl;

    return 0;
}
```



Загальний вигляд команди *зчитування даних з бінарного файлу*:

```
Файлова_змінна.read(адреса_області_пам'яті, sizeof(область_пам'яті));
```

де:

- Файлова_змінна – файлова змінна, пов'язана із відкритим бінарним файлом;
- адреса_області_пам'яті – вказівник на змінну (адреса змінної), значення якої буде зчитане з файлу;
- sizeof(область_пам'яті) – розмір змінної – тої області пам'яті, яка буде заповнена зчитаними з файлу даними; це ж – і розмір зчитаних з файлу даних.

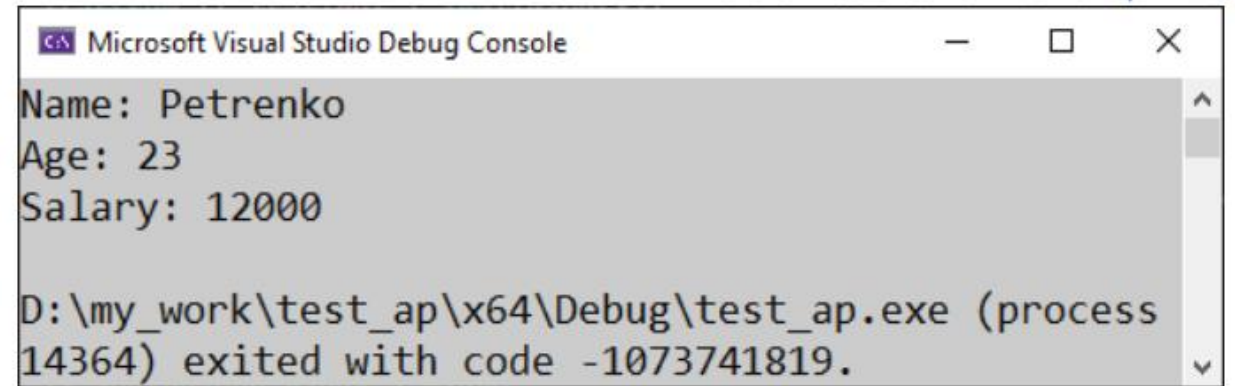
Дані зчитуються з тої компоненти файлу, яка позначена позицією поточного значення файлового вказівника зчитування. *При відкриванні файлу вказівник зчитування встановлюється на початок файлу.*

Оскільки потік визначається як послідовність символів, що передаються від одного довільного пристрою до іншого, то для зчитування даних із бінарного файлу необхідно привести адресу змінної (якщо це – не символьна змінна) до типу «вказівник на символ:


```
#include <iostream>
#include <string>
#include <fstream>
using namespace std;

struct Employee
{
    string name;
    int age;
    float salary;
};

int main(){
    Employee worker;
    // Відкриття файлу для зчитування
    ifstream f("test.dat", ios::binary);
    // Перевірка, чи файл успішно відкрито
    if (!f.is_open()) {
        cout << "failed to open!" << endl;
        return 1;
    }
    // Зчитування даних з файлу
    f.read((char*)&worker, sizeof(Employee));
    // Виведення даних
    cout << "Name: " << worker.name << endl;
    cout << "Age: " << worker.age << endl;
    cout << "Salary: " << worker.salary << endl;
    f.close();
    return 0;
}
```



Microsoft Visual Studio Debug Console

```
Name: Petrenko
Age: 23
Salary: 12000

D:\my_work\test_ap\x64\Debug\test_ap.exe (process
14364) exited with code -1073741819.
```

Для перевірки, чи зчитування відбулось, можна використати умову

```
if (f.fail()) {
    cout << "Помилка запису в файл!" << endl;
    return 1;
}
```

Дякую за увагу

Лектор:

кандидат фіз.-мат. наук, доцент

Шаклеїна Ірина

iryna.o.shakleina@lpnu.ua

кафедра ІСМ, ІКНІ