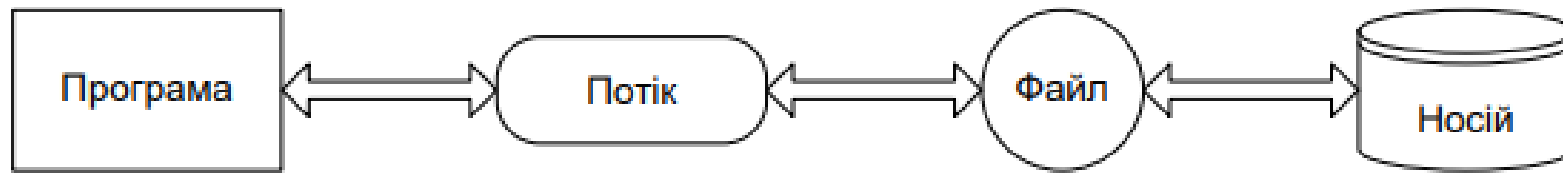




ОПРАЦЮВАННЯ ФАЙЛІВ ЗАСОБАМИ C/C++

Файлова система мови C/C++ призначена для роботи із різноманітними пристроями (зокрема, терміналами, дисководами тощо). Ці фізичні пристрої сильно відрізняються один від іншого, проте буферизована файлова система представляє їх у вигляді логічного пристрою, який називається *поток*.



Текстовий потік – це послідовність символів. Згідно стандарту C, текстовий потік організований у вигляді рядків, кожний з яких закінчується символом нового рядка; в кінці останнього рядка цей символ – не обов’язковий.

можже бути
перетвор. шм-в

Двійковий потік – це послідовність байтів, яка взаємно однозначно відповідає байтам на зовнішньому носії, причому ніяких перетворень символів не відбувається.

к-сть символів
– не змін.

Файли

В мові C *файлом* може бути все що завгодно, починаючи з дискового файлу і закінчуючи терміналом чи принтером.

Потік пов'язують із певним файлом, виконуючи *операцію відкриття*. Як тільки файл відкрито, можна виконувати обмін інформацією між ним та програмою.

При зчитуванні з файлу (або запису до файлу) кожного символу вказівник поточної позиції збільшується, забезпечуючи тим самим просування по файлу

Файл від'єднується від потоку за допомогою операції закриття.

Якщо програма завершується нормально (тобто, коли функція main() повертає управління операційній системі, або коли викликається функція exit()), то всі файли закриваються автоматично.

У випадку аварійного завершення програми файли не закриваються.

Блок управління файлом – невелика область пам’яті, яка виділяється операційною системою для зберігання інформації про відкритий файл, і зазвичай містить інформацію про ідентифікатор файлу, його місце розміщення на диску та вказівник поточної позиції у файлі.

Основна мета розмежування файлів та потоків – забезпечити єдиний *інтерфейс*. Для виконання всіх операцій введення-виведення використовуються потоки, які не залежать від реалізації файлів.

Основи файлової системи

Файлова система мови C/C++ складається із набору пов'язаних між собою функцій.



Деякі функції файлової системи C	
Ім'я	Призначення
fopen()	Відкриває файл
fclose()	Закриває файл
putc()	Записує символ у файл
fputc()	Те ж саме, що і putc()
getc()	Зчитує символ з файлу
fgetc()	Те ж саме, що і getc()
fgets()	Зчитує рядок з файлу
fputs()	Записує рядок у файл
fseek()	Встановлює вказівник поточної позиції на певний байт файлу
ftell()	Повертає значення вказівника поточної позиції у файлі
fprintf()	Для файлу те саме, що printf() для консолі
fscanf()	Для файлу те саме, що scanf() для консолі
feof()	Повертає значення true (істина), якщо досягнуто кінець файлу
ferror()	Повертає значення true , якщо трапилася помилка
rewind()	Встановлює вказівник поточної позиції на початок файлу
rename()	Перейменовує фізичний файл
remove()	Знищує фізичний файл
fflush()	Допише потік у файл (примусово звільняє буфер)

Файлова змінна – вказівник на структуру *FILE*

Файлова змінна – це те, що поєднує в єдине ціле усю систему введення-виведення мови C/C++.

Файлова змінна визначає конкретний файл і використовується відповідним потоком при виконанні операцій введення-виведення: для виконання над файлами операцій зчитування і запису програми мають використовувати відповідні файлові змінні, пов'язані з цими файлами.

Оголошення файлової змінної має такий загальний вигляд: **FILE* f;**

Відкриття файлу

```
FILE *fopen(const char *ім'я-файлу, const char *режим);
```

де

- ім'я-файлу – вказівник на нуль-термінальний літерний рядок, який описує допустиме в операційній системі ім'я файлу (і може містити шлях до файлу);
- режим – вказівник на нуль-термінальний літерний рядок, що описує спосіб, яким буде відкрито цей файл.

Допустимі значення режимів відкриття файлу

Режим	Призначення
r	Відкрити текстовий файл для зчитування
w	Створити текстовий файл для запису
a	Відкрити текстовий файл для додавання даних в його кінець
rb	Відкрити двійковий файл для зчитування
wb	Створити двійковий файл для запису
ab	Відкрити двійковий файл для додавання даних в його кінець
r+	Відкрити текстовий файл для зчитування / запису
w+	Створити текстовий файл для зчитування / запису
a+	Відкрити текстовий файл для додавання даних в його кінець або створити текстовий файл для зчитування / запису
r+b	Відкрити двійковий файл для зчитування / запису
w+b	Створити двійковий файл для зчитування / запису
a+b	Відкрити двійковий файл для додавання даних в його кінець або створити двійковий файл для зчитування / запису

Закриття файлу

Функція `fclose()` закриває потік, відкритий за допомогою `fopen()`. При цьому: у файл записуються всі дані, які ще залишалися в дисковому буфері; звільняється файлова змінна – блок управління файлом, пов'язаний з цим потоком; файл закривається на рівні операційної системи.

Якщо не закрити файл, то це приведе до втрати даних, пошкодження файлів на рівні операційної системи та можливих помилок при виконанні програми.

Прототип функції:

```
int fclose(FILE *файлова-змінна);
```

Опрацювання текстових файлів

Запис та зчитування символу

Запис символу – функції `putc()` та `fputc()`

```
int putc(int ch, FILE *файлова-змінна);
```

файлова-змінна – вказівник на блок управління файлу, отриманий в результаті виклику функції `fopen()`;

ch – символ, що записується у файл.

Зчитування символу – функції `getc()` та `fgetc()`

Для зчитування символів є також дві (для збереження сумісності із попередніми версіями C) повністю еквівалентних функції: `getc()` та `fgetc()`.

```
int getc(FILE *файлова-змінна);  
int fgetc(FILE *файлова-змінна);
```

```
do { ch = getc(f); }  
while ( ch != EOF );
```

Перевірка кінця файлу – функція `feof()`

Для перевірки, чи досягнуто кінець файлу, використовувати функцію `feof()`.

```
int feof(FILE *файлова-змінна);
```

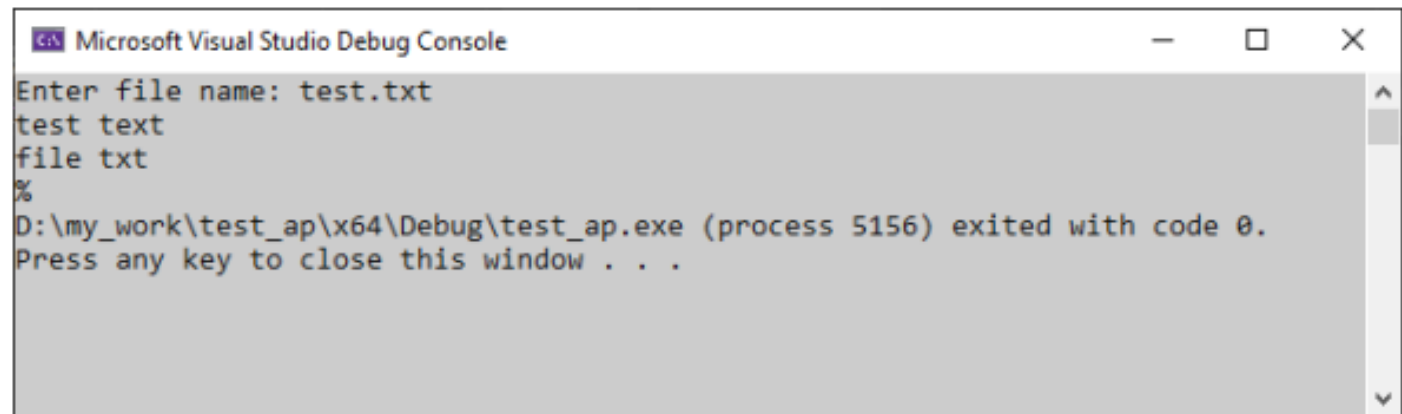
Функція повертає ненульове значення `true` (істина) лише при спробі виконати зчитування після того, як буде досягнуто кінець файлу, та нульове значення `false` (хибність) в іншому випадку.

Приклад використання функцій `fopen()`, `getc()`, `putc()` та `fclose()`



```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
    FILE* f;
    char ch;
    char fname[61];
    cout << "Enter file name: ";
    cin.getline(fname, sizeof(fname));
    if ((f = fopen(fname, "w")) == NULL)
    {
        cerr << "Error opening file '" << fname << "'" << endl;
        exit(1);
    }
    cout << "Enter your text (enter '%' to cancel): " << endl;
    do {
        ch = getchar();
        putc(ch, f);
    } while (ch != '%');
    fclose(f);
    return 0;
}
```

```
#include <iostream>
#include <stdio.h>
using namespace std;
int main()
{
    FILE* f;
    char ch;
    const int LEN = 61;
    char fname[LEN];
    cout << "Enter file name: ";
    cin.getline(fname, LEN);
    if ((f = fopen(fname, "r")) == NULL)
    {
        cerr << "Error opening file " << fname << "" << endl;
        exit(1);
    }
    ch = getc(f);
    while (ch != EOF)
    {
        cout << ch;
        ch = getc(f);
    }
    fclose(f);
    return 0;
}
```



The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads "Microsoft Visual Studio Debug Console". The console output is as follows:

```
Enter file name: test.txt
test text
file txt
%
D:\my_work\test_ap\x64\Debug\test_ap.exe (process 5156) exited with code 0.
Press any key to close this window . . .
```



```
int main()
{
    FILE* in, * out;
    char inName[61], outName[61];
    cout << "Enter input file name: ";
    cin.getline(inName, sizeof(inName));
    if ((in = fopen(inName, "rb")) == NULL)
    {
        cerr << "Error opening input file" << endl;
        exit(1);
    }
    cout << "Enter output file name: ";
    cin.getline(outName, sizeof(outName));
    if ((out = fopen(outName, "wb")) == NULL)
    {
        cerr << "Error opening output file" << endl;
        exit(1);
    }
    char ch;
    // Цей код копіює файл
    while (!feof(in))
    {
        ch = getc(in);
        if (!feof(in))

            putc(ch, out);
    }
    fclose(in);
    fclose(out);
    return 0;
}
```

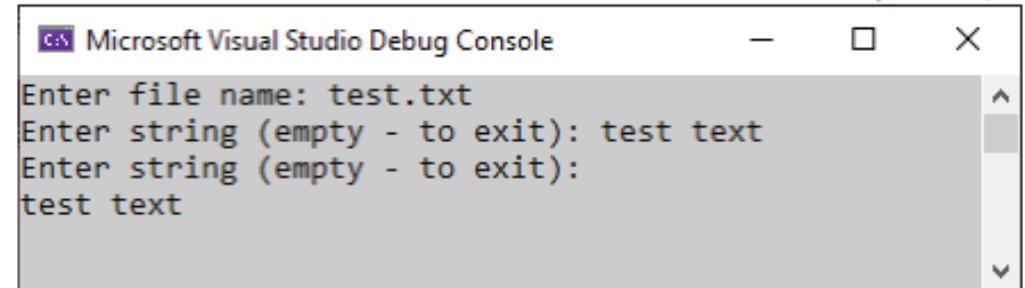

Функція *rewind()*

ВСТАНОВЛЮЄ ВКАЗІВНИК ПОТОЧНОЇ ПОЗИЦІЇ НА ПОЧАТОК ФАЙЛУ

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main()
{
```

```
int rewind(FILE *файлова-змінна);
```

```
    const int LEN = 80;
    char s[LEN];
    FILE* f;
    char fname[61];
    cout << "Enter file name: "; cin.getline(fname, sizeof(fname));
    if ((f = fopen(fname, "w+")) == NULL)
    {
        cerr << "Error opening file " << fname << "" << endl;
        exit(1);
    }
    do {
        cout << "Enter string (empty - to exit): ";
        cin.getline(s, LEN - 1); // на 1 менше, щоб було місце для '\n'
        strcat(s, "\n"); // додавання символу нового рядка
        fputs(s, f);
    } while (*s != '\n');
    rewind(f); // "перемотали" файл до початку
    while (!feof(f)) // поки не кінець файлу
    {
        fgets(s, LEN - 1, f); // зчитуємо файл і
        cout << s; // виводимо його вміст на екран
    }
    return 0;
}
```



```
Microsoft Visual Studio Debug Console
Enter file name: test.txt
Enter string (empty - to exit): test text
Enter string (empty - to exit):
test text
```

Опрацювання бінарних файлів

Функції *fread()* та *fwrite()*

дозволяють зчитувати та записувати блоки даних будь-якого типу.

```
size_t fread(void *буфер, size_t розмір, size_t кількість, FILE *фз);
```

```
size_t fwrite(const void *буфер, size_t розмір, size_t кількість, FILE *фз);
```

де буфер – вказівник на елемент даних:

для функції *fread()* – область пам'яті, в яку будуть записані дані, прочитані з файлу;

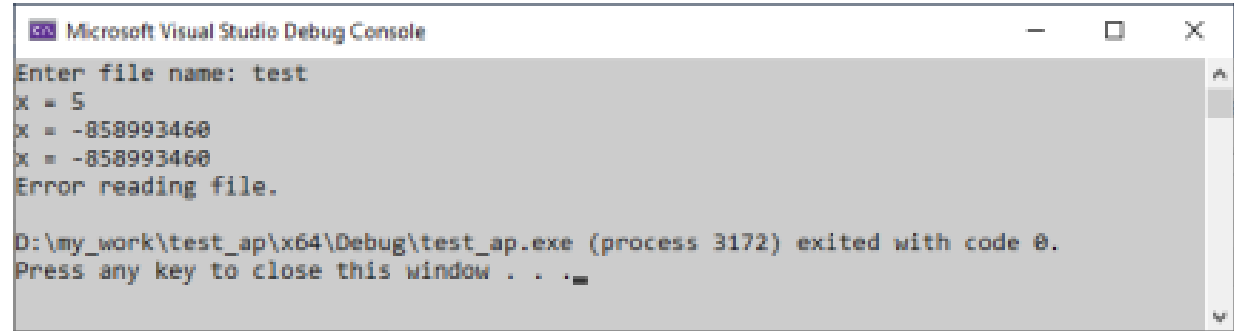
для функції *fwrite()* – вказівник на область пам'яті, яка містить дані, що будуть записані у файл; розмір – довжина одного елементу даних в байтах (розмір області пам'яті, на яку налаштований вказівник буфер);

кількість – визначає, скільки елементів даних буде прочитано (*fread*) чи записано (*fwrite*);

фз – файлова змінна – вказівник на блок управління файлом, пов'язаний із вже відкритим потоком.

```
#include <iostream>
#include <stdio.h>
using namespace std;
```

```
int main()
{
    char fname[61];
    cout << "Enter file name: "; cin >> fname;
    FILE* f;
    f = fopen(fname, "wb");
    int x = 5;
    fwrite(&x, sizeof(x), 3, f); // 3 - це не кількість екземплярів змінної x:
    // буде зчитано три значення типу int та
    // записано у файл
    fclose(f);
    f = fopen(fname, "rb");
    while (!feof(f))
    {
        if (fread(&x, sizeof(x), 1, f) != 1)
            if (feof(f))
            {
                cerr << "Error reading file." << endl;
                break;
            }
        cout << "x = " << x << endl;
    }
    fclose(f);
    return 0;
}
```



```
Microsoft Visual Studio Debug Console

Enter file name: test
x = 5
x = -858993468
x = -858993468
Error reading file.

D:\my_work\test_ap\x64\Debug\test_ap.exe (process 3172) exited with code 0.
Press any key to close this window . . .
```



Прямий доступ: функції *fseek()* та *ftell()*

Функція *fseek()* встановлює вказівник поточної позиції.

```
int fseek(FILE *файлова-змінна, long кількість-байт, int початок-відліку);
```

де файлова-змінна – вказівник на блок управління файлом, результат виклику функції *fopen()*;

кількість-байт – кількість байт, відносно початку-відліку,

початок-відліку – одне із наступних трьох значень:

Початок відліку	Значення
Початок файлу	SEEK_SET
Поточна позиція	SEEK_CUR
Кінець файлу	SEEK_END

При успішному завершенні функція *fseek()* повертає значення 0.

Стандартні потоки

Для програми на мові C, на початку її виконання автоматично відкриваються три потоки:

- `stdin` – стандартний потік введення,
- `stdout` – стандартний потік виведення
- `stderr` – стандартний потік помилок.

Зазвичай ці потоки пов'язані з консоллю, проте їх можна перенаправити на інший пристрій.

Перенаправлення потоків

Для перенаправлення потоків можна використати функцію `freopen()`. Вона пов'язує потік з новим файлом. Її прототип:

```
FILE *freopen(const char *ім'я-файлу, const char *режим, FILE *потік);
```

де

`ім'я-файлу` – вказівник на рядок, що містить ім'я файлу, який потрібно пов'язати з потоком, на який вказує вказівник потік.

Як і функція `fopen()`, якщо функція `freopen()` виконалася успішно, то вона повертає потік, якщо ж були помилки, – то значення `NULL`.

Дякую за увагу

Лектор:
кандидат фіз.-мат. наук, доцент
Шаклеїна Ірина
iryna.o.shakleina@lpnu.ua
кафедра ІСМ, ІКНІ