

Введення та виведення. Основи роботи з файлами.

*Вред или польза действия обуславливается
совокупностью обстоятельств.*

Козьма Прутков

*

Введення та виведення

Мова C++ підтримує дві системи введення/виведення. Перша – спадок C. 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, можливість перевизначення для власних класів. Але вона вимагає розуміння об'єктів та класів.

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

Почнемо з введення/виведення в “стилі C” .

Введення та виведення

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.

Наприклад:

- `getchar()`, `putchar()` – читання та запис символу;
- `gets()`, `puts()` – читання та запис рядка;
- `scanf()`, `printf()` – форматоване введення та виведення даних

Завдання

Розібратися з переліченими функціями: `getchar()`, `putchar()`, `gets()`, `puts()`, `scanf()`, `printf()`, з форматним рядком для функцій `scanf()`, `printf()`, з специфікаторами форматів.

Приклад

```
#include <stdio>
#include <ctype>
//приведення символів до нижнього регістру

int main() {
    int c;
    while ((c=getchar()) != '\n')
        putchar(isupper(c) ? tolower(c) : c);
    system("pause"); return 0;
}
```

Приклад

```
#include <stdio>
//простий калькулятор

int main() {
    double sum = 0, v;
    printf("Enter numbers.  Finish - <Enter+Ctrl+Z>\n");
    while (scanf("%lf", &v) != EOF)
        printf("\t%.3f\n", sum += v);
    system("pause"); return 0;
}
```

Обробка файлів

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.

Потоки бувають двох типів: текстові та двійкові.

Файловий покажчик

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 `<stdio>` (та `<stdio.h>`).

Відкривання потоків

Потік можна відкрити для читання або/та запису в текстовому або двійковому режимі. Функція відкриття потоку має формат:

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

Якщо відкривання було успішним, функція повертає *показчик файлу*, що містить всю необхідну для роботи з потоком інформацію; інакше функція повертає **NULL**.

Параметр `filename` – задає ім'я файлу у вигляді рядка в стилі *C*.

Параметр `mode` – визначає режим відкривання файлу.

Режими відкривання файлу

- “r” – відкриття існуючого файлу для читання;
- “w” – створення нового файлу для запису (якщо файл з таким ім`ям існує – він перезаписується);
- “a” – відкриття існуючого файлу для додавання в його кінець нової інформації;
- “r+” – відкриття існуючого файлу для читання й запису;
- “w+” – створення нового файлу для читання й запису (перезаписується файл з таким ім`ям існує, якщо існує);
- “a+” – відкриття існуючого файлу для читання та додавання в його кінець нової інформації.

Режим може містити символи **t** – текстовий, або **b** – двійковий. По умовчанням передбачається текстовий режим (**t**).

Приклади

```
FILE *fp;
```

```
fp = fopen("file1.txt", "a+");
```

```
FILE *f = fopen("c:\\temp\\data.dat", "rb+");
```

```
FILE *fp;
```

```
if (fp = fopen("file2.txt", "r") == NULL)
```

```
{perror("ERROR open file2.txt"); return 1;}
```

```
FILE *fp = fopen("./data.dat", "w");
```

```
FILE *fp1 = fopen("./data1.dat", "w");
```

Закривання потоків

Потік закривається або при завершенні програми, або явно функцією `fclose`:

```
int fclose(FILE * );
```

Якщо потік, заданий параметром-показчиком файлу, був відкритий для запису, то перед закриттям у файл записуються данні, що містяться у буферах потоку. У випадку успішного закриття функція повертає `0`, інакше – `EOF`.

Рекомендується завжди явно закривати потоки, що відкриті для запису, щоб уникнути втрати даних.

```
int fflush(FILE * );
```

 - примусово скидає буфер у файл.

Максимальна кількість одночасно відкритих файлів визначається макросом `FOPEN_MAX`.

Введення та виведення

Можна здійснювати у вигляді послідовності байтів, символів, рядків, або з використанням форматних перетворень. Існують відповідні функції бібліотеки.

Операції введення/виведення завжди виконуються починаючи з поточної позиції потоку, що визначається *показчиком потоку*. *Показчик потоку* встановлюється при відкритті на початок, або кінець (в залежності від режиму відкриття) й змінюється автоматично при виконанні операцій введення/виведення. Поточне положення *показчика потоку* можна отримати функціями `ftell`, `fgetpos` й задати функціями `fseek`, `fsetpos`. Ці функції не можна використовувати для *стандартних потоків*.

Основні функції введення/виведення

- читання символу – `fgetc`, `(getc)`, `(getchar - stdin)`;
- запис символу – `fputc`, `(putc)`, `(putchar - stdout)`;
- читання рядка - `fgets`, `(gets - stdin)`;
- запис рядка - `fputs`, `(puts - stdout)`;
- форматоване читання – `fscanf` , `(scanf - stdin)`;
- форматований запис – `fprintf` , `(printf - stdout)`;
- читання та запис потоку байтів – `fread`, `fwrite`.

Читання й запис символів та рядків

`fputc()`, `putc()` – записують один символ. У випадку успіху повертають значення символу, інакше – EOF. *Формат:*

```
int fputc(int ch, FILE *fp);  int putc(int ch, FILE *fp);
```

`fgetc()`, `getc()` – читають один символ. У випадку успіху повертають значення символу, інакше – EOF. *Формат:*

```
int fgetc(FILE *fp);  int getc(FILE *fp);
```

`fputs()` – записує рядок. Повертає у випадку успіху невід'ємне число, інакше – EOF. (символ кінця не записує).

`fgets()` – читає один рядок (але не більш ніж 2 параметр-1).

Формат:

```
int fputs(const char *s, FILE *fp);
```

```
char *fgets(char *s, int n, FILE *fp);
```


Форматоване читання й запис

`fprintf` – форматований запис. За наявності помилки повертає – EOF. *Формат:*

```
fprintf(FILE *fp, const char *format [,arg1, ...]);
```

`fscanf`– форматоване читання. Як результат повертає кількість прочитаних й перетворених даних. *Формат:*

```
fscanf(FILE *fp, const char *format [,adr1, ...]);
```

`const char *format` – рядок спеціального формату, дозволяє вказувати звичайні символи та специфікатори формату, що починаються %.

Приклади:

```
fprintf(fp, “%d %d”, a, b);
```

```
fscanf(fp, “%d”, &x);
```


Запис потоку байтів

`fwrite` – записує об'єкт у файл, відкритий у двійковому режимі. Функція повертає кількість записаних об'єктів.

Формат:

```
size_t fwrite(const void *Buf, size_t Size, size_t Count,  
FILE *fp);
```

`Buf` – покажчик на об'єкт;

`Size` – розмір об'єкту у байтах;

`Count` – кількість об'єктів, що записуються;

`fp` – файловий-покажчик.

Приклад

```
#include <cstdio>
#include <iostream>
using namespace std;
int main() {
    FILE *fp = fopen("d:\\Tmp\\data1.dat", "wb");
    struct Point {
        int x, y;
    } point;
    point.x = 10; point.y = 20;
    int x = fwrite(&point, sizeof(Point), 1, fp);
    cout << x << endl;
    system("pause"); return 0;
}
```

Читання потоку байтів

`fread` – зчитує об'єкт з файлу, відкритого у двійковому режимі. Функція повертає кількість зчитаних об'єктів.

Формат:

```
size_t fread(void *Buf, size_t Size, size_t Count, FILE *fp);
```

`Buf` – покажчик на об'єкт;

`Size` – розмір об'єкту у байтах;

`Count` – кількість об'єктів, що записуються;

`fp` – файловий-покажчик.

Обробка помилок

Функції роботи з потоком повертають значення, які потрібно аналізувати й обробляти помилкові ситуації. Крім того використовують функції:

`int feof(FILE *)` – повертає ненульове значення, якщо досягнутий кінець файлу й 0 у протилежному випадку.

`int ferror(FILE *)` – повертає ненульове значення, якщо виявлена помилка введення/виведення й 0 у протилежному випадку.

До функцій потрібно звертатись відразу після виконання операцій з файлом.

Функції для роботи з буферним покажчиком

Дозволяють здійснювати читання та запис у файл з довільної позиції. Використовуються в основному для файлів відкритих у двійковому режимі.

`long ftell(FILE *fp)` – повертає поточну позицію покажчика (при помилка `-1L`).

`int fgetpos(FILE *fp, fpos_t *pos)` – записує поточну позицію покажчика у другий параметр, повертає `0`, якщо помилок не має, інакше ненульове значення.

`void rewind(FILE *fp)` – встановлює покажчик на початок файлу.

`int fsetpos(FILE *fp, const fpos_t *pos)` – встановлює покажчик за параметром `pos`, повертає `0`, якщо помилок не має, інакше ненульове значення.

Функції для роботи з буферним покажчиком

`int fseek(FILE *fp, long offset, int origin)` – встановлює покажчик згідно зміщення `offset` відносно позиції `origin`.

Для параметру `origin` можуть бути вказані макроси:

`SEEK_SET` – початок файлу;

`SEEK_CUR` – поточна позиція покажчика;

`SEEK_END` – кінець файлу.

Приклад

У файлі зберігаються данні про монітори. Кожний рядок містить: тип монітору (20 симв.), ціни – оптова (10 симв.) та роздрібна (10 симв.), примітка (30 симв.).

Зчитуючи данні з текстового файлу, сформувати відповідну структуру й записати в двійковому режимі у вихідний файл.

Демонструється можливість читання довільного запису з двійкового файлу.

Видалення файлу

```
int remove(const char *filename);
```

Функція видаляє файл `filename` з файлової системи за умови, що він існує. У випадку успіху повертає `0`, інакше `-1` і встановлює значення змінної `errno` рівним `ENOENT` (файл не знайдено), або `EACCES` (доступ заборонено).

Перенаправлення потоків

На початку виконання програми автоматично відкриваються три стандартних потоки: `stdin` (введення), `stdout` (виведення), `stderr` (помилки). Ці ідентифікатори є файловими покажчиками, що зв'язані за умовчанням з вікном консолі. Їм не можна присвоювати значення, не потрібно закривати наприкінці роботи, але можна вказувати у функціях призначених для роботи з файлами. *Наприклад:*

```
fputs("String1\nString2", stdout);
```

```
fflush(stdout);
```

Стандартні потоки можна перенаправити, щоб дані зчитувались/записувались з файлу. Для цього існує два способи.

Перенаправлення потоків

з командного рядка

Не вимагає змін у програмі. У командному рядку можна скористатись однією з команд:

`test.exe > file.txt` – результат виконання програми `test.exe` записується у файл `file.txt`. Якщо такого файлу не існує – він створюється, інакше – перезаписується.

`test.exe >> file.txt` – відбувається дозаписування в кінець файлу `file.txt` .

`test.exe < file.txt` – використовується для введення даних з файлу `file.txt` .

Можна поєднувати:

`test.exe < file.txt1 > file.txt2`

Перенаправлення потоків за допомогою функції `freopen()`

Вимагає підключення `#include <stdio>` . *Формат:*

```
FILE *freopen(const char *Filename, const char *Mode,  
FILE *fp);
```

`Filename`, `Mode` – ідентичні параметрам `fopen()` . Якщо виникли помилки – повертає `NULL`.

Приклад:

```
FILE *fp = 0;  
fp = freopen("file.txt", "w", stdout);  
if (!fp) exit (1);  
printf("%s\n", "Записуємо у файл");
```

Зауваження

- Перенаправлення стандартних потоків може бути корисним, наприклад, при налагодженні й тестуванні. Однак виконання дискових операцій введення/виведення на перенаправлених потоках є менш ефективним у порівнянні з традиційними операціями з файлами.

Підсумки

- Розглянули основні можливості для введення/виведення в “стилі C”.

Поради

- Розібратися з розглянутими можливостями, відповідними функціями стандартної бібліотеки, форматним рядком та специфікаторами перетворень.
- Явно й вчасно закривати відкриті потоки.
- Потрібно уважно використовувати можливості позиціонування на запис.
- Потрібно використовувати контроль та обробку можливих помилкових ситуацій.

Задачі

- Написати функцію для дописування до елементів першого файлу елементів другого зі зберіганням результату в першому файлі.
- Множину цілих представлено файлом, дані якого упорядковано за зростанням. За двома такими файлами створити третій файл, який також є упорядкованим і подає їх
 - а) об'єднання;
 - б) перетин;
 - в) різницю;
 - г) симетричну різницю.
- Порівняти два текстові файли. Надрукувати перший рядок в якому вони відрізняються.

Задачі

- F файл, що містить цілі числа. Записати в G всі парні, а в H всі непарні числа з збереженням порядку.
- Написати програму підрахунку числа рядків у тексті.
- Є текстовий файл та рядок S. Записати у інший файл всі його рядки, що містять як фрагмент S.
- Написати програму для пошуку у текстовому файлі вказаного слова. Слово та ім'я файлу подаються у командному рядку.
- Для текстового файлу f прибрати всі пропуски, що розташовані в кінці рядків. Результат отримати у файлі g.
- Написати функцію для копіювання текстового файлу.

Задачі

- Файл містить інформацію про книжки (автор, назва, видавництво, рік, обсяг).
 - Знайти назви книг зазначеного автора, виданих з 2000 р.;
 - Відібрати книжки про програмування;
 - Підрахувати загальний обсяг по видавництву.