

## Лабораторна робота №11

### Файловий ввід / вивід

Мета роботи: вивчити програмні засоби для роботи з файлами та потоками; дослідити основні функції роботи з файлами та реалізовувати найпростіші операції з ними; навчитись застосовувати у своїх програмах вхідні і вихідні текстові файли і файлові потоки.

### Теоретичні відомості

Є три основні класи файлового введення / виводу в C ++ : **ifstream**, **ofstream**; **fstream**.

За допомогою цих класів можна виконувати односпрямований файловий ввід, односпрямований файловий висновок і двонаправлений файловий ввід / вивід. Для їх використання потрібно всього лише підключити заголовний файл `fstream`.

На відміну від потоків `cout`, `cin`, `cerr` і `clog`, які відразу ж можна використовувати, файлові потоки повинні бути явно встановлені програмістом. Тобто, щоб відкрити файл для читання і / або запису, потрібно створити об'єкт відповідного класу файлового введення / виводу, вказавши ім'я файлу в якості параметра. Потім, за допомогою операторів вставки (`<<`) або видалення (`>>`), можна записувати дані в файл або читати вміст файлу. Після цього фінал - потрібно закрити файл: явно викликати метод `close ()`.

### Файловий вивід

Для запису в файл використовується клас `ofstream`, наприклад:

```
1 #include <iostream>
2 #include <fstream>
3 #include <cstdlib> // для использования exit()
4
5 int main()
6 {
7     using namespace std;
8
9     // ofstream используется для записи данных в файл
10    // Создаём файл SomeText.txt
11    ofstream outf("SomeText.txt");
12
13    // Если мы не можем открыть этот файл для записи данных в него
14    if (!outf)
15    {
16        // То выводим сообщение об ошибке и выполняем exit()
17        cerr << "Uh oh, SomeText.txt could not be opened for writing!" << endl;
18        exit(1);
19    }
20
21    // Записываем в файл следующие две строчки
22    outf << "See line #1!" << endl;
23    outf << "See line #2!" << endl;
24
25    return 0;
26
27    // Когда outf выйдет из области видимости, то деструктор класса ofstream автоматически закроет
28 }
```

## Файловий ввід

Спробуємо прочитати вміст файлу, який створили в попередньому прикладі. Зверніть увагу, `ifstream` повертає 0, якщо ми досягли кінця файлу (це зручно для визначення «довжини» вмісту файлу), наприклад:

```
1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 #include <cstdlib> // для использования exit()
5
6 int main()
7 {
8     using namespace std;
9
10    // ifstream используется для чтения содержимого файла
11    // Попытаемся прочитать содержимое файла SomeText.txt
12    ifstream inf("SomeText.txt");
13
14    // Если мы не можем открыть этот файл для чтения его содержимого
15    if (!inf)
16    {
17        // То выводим следующее сообщение об ошибке и выполняем exit()
18        cerr << "Uh oh, SomeText.txt could not be opened for reading!" << endl;
19        exit(1);
20    }
21
22    // Пока есть данные, которые мы можем прочитать
23    while (inf)
24    {
25        // То перемещаем эти данные в строку, которую затем выводим на экран
26        string strInput;
27        inf >> strInput;
28        cout << strInput << endl;
29    }
30
31    return 0;
32
33    // Когда inf выйдет из области видимости, то деструктор класса ifstream автоматически закроет н
34 }
```

Оператор вилучення працює з «відформатовані даними», тобто він ігнорує всі пропуски, символи табуляції і символ нового рядка. Щоб прочитати весь вміст як є, без його розбивки на частині (як в прикладі вище), нам потрібно використовувати метод **getline ()**:

```

1 #include <iostream>
2 #include <fstream>
3 #include <string>
4 #include <cstdlib> // для використання exit()
5
6 int main()
7 {
8     using namespace std;
9
10    // ifstream використовується для читання змісту файлів
11    // Ми спробуємо прочитати зміст файлу SomeText.txt
12    ifstream inf("SomeText.txt");
13
14    // Якщо ми не можемо відкрити файл для читання його змісту
15    if (!inf)
16    {
17        // То виводимо наступне повідомлення про помилку і виконуємо exit()
18        cerr << "Uh oh, SomeText.txt could not be opened for reading!" << endl;
19        exit(1);
20    }
21
22    // Поки є, що читати
23    while (inf)
24    {
25        // То переміщуємо те, що можемо прочитати, в строку, а потім виводимо цю строку на екран
26        string strInput;
27        getline(inf, strInput);
28        cout << strInput << endl;
29    }
30
31    return 0;
32
33    // Коли inf вийде з області видимості, то деструктор класу ifstream автоматично закриє н
34 }

```

## Режими відкриття файлів

Що станеться, якщо ми спробуємо записати дані в уже існуючий файл? Повторний запуск програми вище (найперша) показує, що вихідний файл повністю перезаписується при повторному запуску програми. А що, якщо нам потрібно додати дані в кінець файлу? Виявляється, конструктори файлового потоку приймають необов'язковий другий параметр, який дозволяє вказати програмісту спосіб відкриття файлу. Можна передавати такі прапори (які знаходяться в класі ios):

- app - відкриває файл в режимі додавання;
- ate - переходить в кінець файлу перед читанням / записом;
- binary - відкриває файл в бінарному режимі (замість текстового режиму);
- in - відкриває файл в режимі читання (за замовчуванням для ifstream);
- out - відкриває файл в режимі запису (за замовчуванням для ofstream);
- trunc - видаляє файл, якщо він вже існує.

Можна вказати відразу кілька прапорів шляхом використання побітового АБО (|).

ifstream за замовчуванням працює в режимі ios :: in;  
ofstream за замовчуванням працює в режимі ios :: out;  
fstream за замовчуванням працює в режимі ios :: in АБО ios :: out, що означає, що ви можете виконувати як читання вмісту файлу, так і запис даних в файл.

Розглянемо програму, яка додасть два рядки в раніше створений файл SomeText.txt:

```
1 #include <iostream>
2 #include <cstdlib> // для использования exit()
3 #include <fstream>
4
5 int main()
6 {
7     using namespace std;
8
9     // Передаём флаг ios::app, чтобы сообщить fstream, что мы собираемся добавить свои данные к уже
10    // мы не собираемся перезаписывать файл. Нам не нужно передавать флаг ios::out,
11    // поскольку ofstream по умолчанию работает в режиме ios::out
12    ofstream outf("SomeText.txt", ios::app);
13
14    // Если мы не можем открыть файл для записи данных
15    if (!outf)
16    {
17        // То выводим следующее сообщение об ошибке и выполняем exit()
18        cerr << "Uh oh, SomeText.txt could not be opened for writing!" << endl;
19        exit(1);
20    }
21
22    outf << "See line #3!" << endl;
23    outf << "See line #4!" << endl;
24
25    return 0;
26
27    // Когда outf выйдет из области видимости, то деструктор класса ofstream автоматически закроет
28 }
```

```
1 #include <fstream>
2
3 int main()
4 {
5     using namespace std;
6
7     ofstream outf("SomeText.txt");
8     outf << "See line #1!" << endl;
9     outf << "See line #2!" << endl;
10    outf.close(); // явно закрываем файл
11
12    // Упс, мы кое-что забыли сделать
13    outf.open("SomeText.txt", ios::app);
14    outf << "See line #3!" << endl;
15    outf.close();
16
17    return 0;
18
19    // Когда outf выйдет из области видимости, то деструктор класса ofstream автоматически закроет
20 }
```

Аналогічно тому як явно закриваємо файл за допомогою методу **close ()**, точно ми можемо і явно відкривати файл за допомогою методу **open ()**. **open ()** працює аналогічно конструкторам класу файлового введення / виводу: приймає ім'я файлу і режим (необов'язково), в якому потрібно відкрити файл, в якості параметрів.

### Порядок виконання роботи

1. Ознайомитись з теоретичним матеріалом, з функціями стандартних бібліотек для роботи файлами і файловими потоками.
2. Дослідити процес реалізації завдань прикладів, відлагодити наведені програму на своєму комп'ютері.
3. Розробити власну програму, які реалізує індивідуальне завдання.
4. Підготувати звіт:
  - варіант і текст завдання;
  - лістинг програми;
  - роздруківку вхідних і вихідних файлів і результати виконання;
  - висновки.

### Варіанти індивідуальних завдань

Вхідні дані вводяться з клавіатури і після введення записувалися у файл; програма має можливість дописувати дані у файл; дані з файлу виводяться на екран.

№	Варіанти індивідуальних завдань
1	NAZV – назва пункту призначення; NUMR – номер потягу; DATE, TIME – дата і час відправлення.
2	NAME – прізвище та ініціали працівника; POS – назва посади; YEAR, MONTH – рік і місяць прийняття на роботу.
3	NAME – назва товару; TYPE – одиниця виміру; COST – ціна одиниці товару; QUANTITY – кількість.
4	CITY – назва населеного пункту призначення; NUM – номер рейса; TYPE – тип літака.
5	NAME – прізвище, ім'я; TEL – номер телефону; BDAY – день народження
6	EXAM – результати вступних іспити з трьох предметів (масив з трьох елементів)
7	FIRST – назва початкового пункту маршруту; FINAL – назва кінцевого пункту маршруту; NUM – номер маршруту; DISTANCE – відстань у кілометрах

8	BEG, END – назви початкового і кінцевого пунктів маршруту; NUM – номер маршруту; DIST – відстань у кілометрах
9	NAZV – назва пункту призначення; NUMR – номер поїзда; DATE – дата відправлення; TIME – час відправлення.
10	NAME – назва товару; TYPE – одиниця виміру товару; SORT – сорт товару; QUANTITY – кількість одиниць товару; COST – ціна

- 1.Алгоритмизация. Введение в язык программирования С++/ И.Е. Белоцерковская, Н.В. Галина, Л.Ю. Катаева - М.: Национальный Открытый Университет "ИНТУИТ", 2016
2. Орленко П. А., Евдокимов П. В. С++ на примерах. Практика, практика и только практика.СПб.: Наука и Техника, 2019. 288 с.
3. Урок №212. Базовый файловый ввод/вывод.URL: <https://ravesli.com/urok-212-bazovyy-fajlovyy-vvod-vyvod/>