

Робота з файлами:

Файл – це деяка послідовність байтів, яка має своє ім'я, наприклад ми будемо розглядати файл з ім'ям test.txt.

Для роботи з файлами необхідно підключити файл заголовку `<fstream>`. В ньому визначено декілька класів і підключені файли заголовків `<ifstream>` — файловий ввід та `<ofstream>` — файловий вивід.

Файловий ввід/вивід проводиться з файлу та у файл.

Наприклад, нам необхідно створити файл і записати фразу «Начинаем работать с файлами».

Для цього необхідно виконати наступне:

1. Створити об'єкт класу `ofstream`;
2. Зв'язати об'єкт класу з файлом, в який ми хочемо записати інформацію;
3. Записати строку в файл;
4. Закрити файл.

```
1  ofstream /* ім'я об'єкту */; // об'єкт класу ofstream
```

Назвемо об'єкт – `fout`, Отримуємо::

```
1  ofstream fout;
```

Для чого нам об'єкт? Він потрібен, щоб було можливо виконати запис у файл. Об'єкт вже створений, але ще не зв'язаний з файлом, в який необхідно записати строку.

```
2  fout.open("test.txt"); // зв'язуємо об'єкт з файлом
```

Через операцію крапка отримуємо доступ до методу класу `open()`, в круглих дужках якого ми повинні вказати ім'я файлу. Цей файл буде створений в поточній директорії з програмою.

Що буде, якщо такий файл існує: він буде замінений новим.

Після відкриття файлу ми можемо записати у нього строку. Робимо таким чином:

```
3  fout << "Начинаем работать с файлами "; // запис строки в файл
```

Оскільки ми більше не будемо змінювати сам файл, ми повинні його закрити.

```
4  fout.close(); // закриваємо файл
```

Підсумок: – файл зі строкою " Начинаем работать с файлами. " створено.

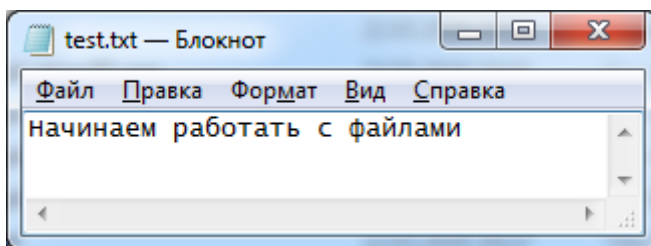
Шаги 1 та 2 можливо поєднати. І ми робимо це таким чином:

- 1 `ofstream fout("test.txt");` // створюємо об'єкт класу `ofstream` та зв'язуємо його з файлом `test.txt`

Поєднуємо всі наші оператори і отримуємо наступне:

```
1    #include <fstream>
2    using namespace std;
3    int main(int argc, char* argv[])
4    {
5        ofstream fout("test.txt");
6        fout << " Начинаем работать с файлами. ";
7        fout.close();
8        system("pause");
9        return 0;
10   }
```

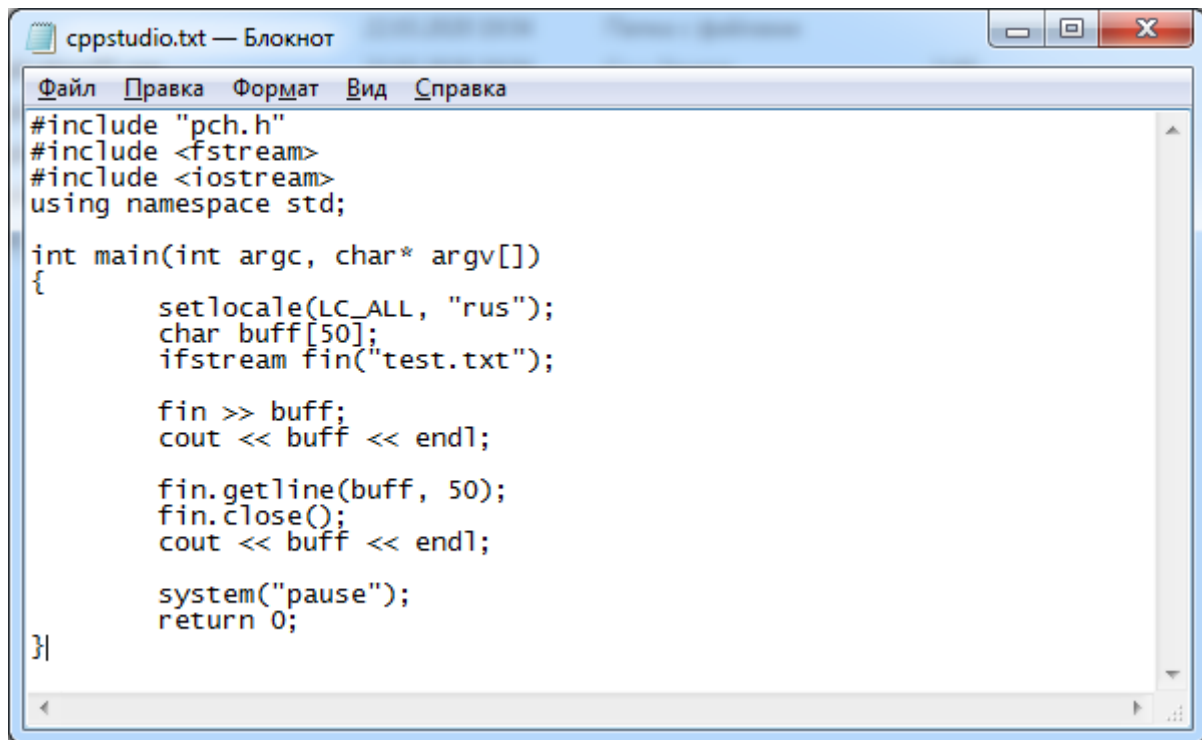
Вихідний файл:



Залишилось перевірити правильність роботи програми, для цього ми відкриваємо його і зчитуємо дані, що в ньому знаходяться

Для читання файлу необхідно виконати наступні кроки:

1. створити об'єкт класу `ifstream` та зв'язати його з файлом, із якого буде проводитись зчитування;
2. прочитати файл;
3. закрити файл.



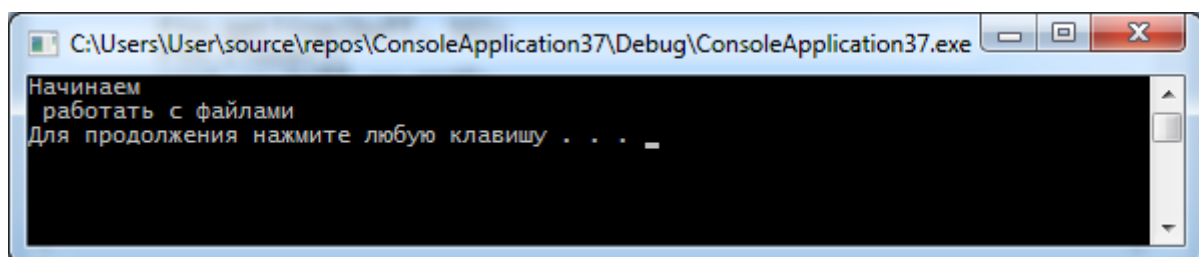
```
#include "pch.h"
#include <fstream>
#include <iostream>
using namespace std;

int main(int argc, char* argv[])
{
    setlocale(LC_ALL, "rus");
    char buff[50];
    ifstream fin("test.txt");

    fin >> buff;
    cout << buff << endl;

    fin.getline(buff, 50);
    fin.close();
    cout << buff << endl;

    system("pause");
    return 0;
}
```

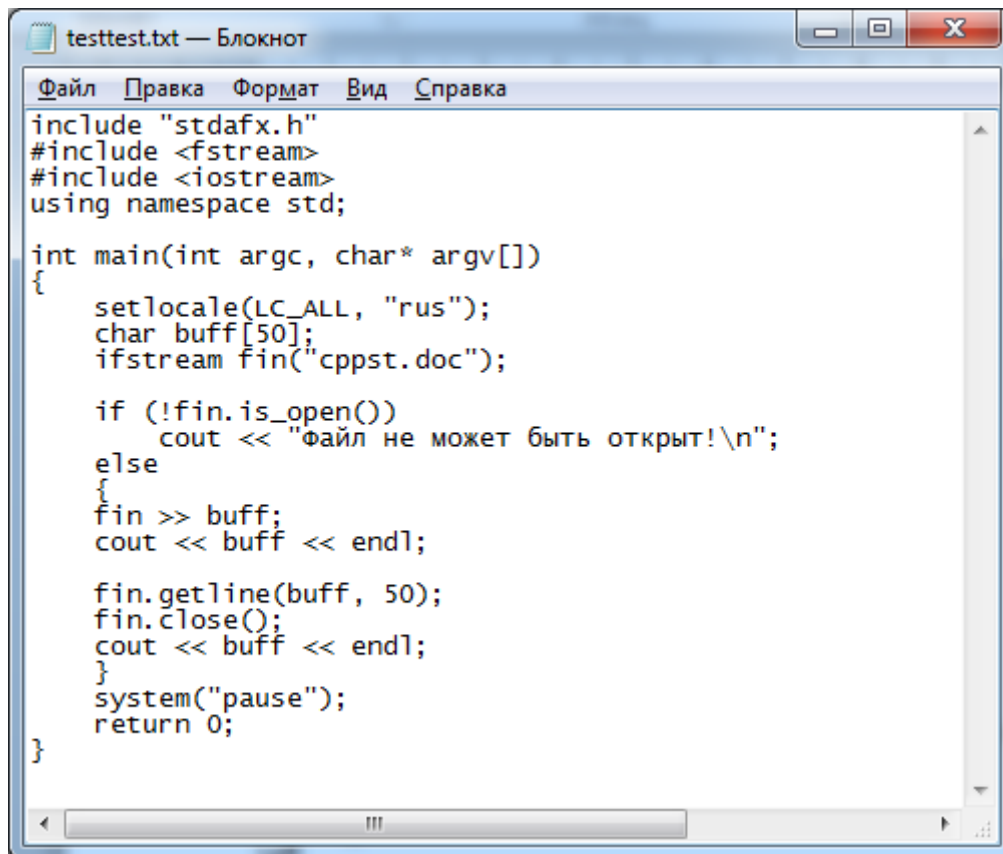


```
Начинаем
работать с файлами
Для продолжения нажмите любую клавишу . . . _
```

У програмі бачимо два шляхи читання із файлу, перший – використання операції передачі у потік, а другий – використання функції `getline()`. В першому випадку зчитали лише перше слово, у другому – строку довжиною у 50 символів. Але в файлі залишилось менше 50 символів, і зчитуються усі символи до останнього. Як бачимо, `fin.getline(buff,50)` продовжило зчитування після першого слова, а не зчитувало дані з початку текстового файлу.

Програма виконала свою роботу, але ,наприклад, в програму передали ім'я файлу, який не існує або в імені файлу є помилка. Що робити? Якщо запустимо нашу програму, то нічого і не буде. Файл не буде знайдений, прочитати його неможливо, і компілятор проігнорує рядки, де виконується робота з файлом. Начебто нормально, але користувачеві не буде зрозуміло, що трапилось. Для цього випадку є функція — `is_open()`, яка повертає цілі значення: 1 —файл успішно відкритий, 0 —файл не був відкритий.

Допрацюємо нашу програму та виведемо необхідне повідомлення.

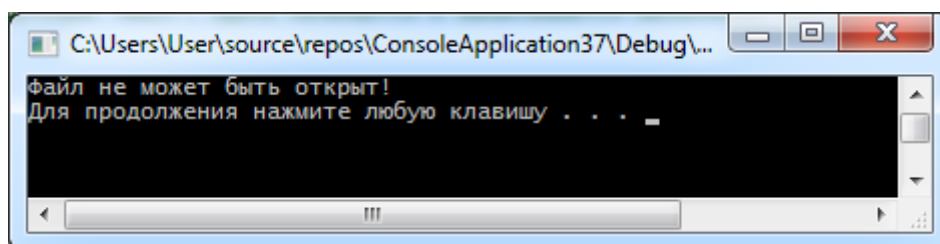


```
testtest.txt — Блокнот
Файл  Правка  Формат  Вид  Справка
#include "stdafx.h"
#include <fstream>
#include <iostream>
using namespace std;

int main(int argc, char* argv[])
{
    setlocale(LC_ALL, "rus");
    char buff[50];
    ifstream fin("cppst.doc");

    if (!fin.is_open())
        cout << "Файл не может быть открыт!\n";
    else
    {
        fin >> buff;
        cout << buff << endl;

        fin.getline(buff, 50);
        fin.close();
        cout << buff << endl;
    }
    system("pause");
    return 0;
}
```



Режими відкриття файлів:

<code>ios_base::in</code>	Відкрити файл для читання
<code>ios_base::out</code>	Відкрити файл для запису
<code>ios_base::ate</code>	При відкритті перемістити покажчик у кінець файлу
<code>ios_base::app</code>	Відкрити файл для запису у кінець файлу
<code>ios_base::trunc</code>	Видалити зміст файлу, якщо він існує
<code>ios_base::binary</code>	Відкриття файлу в двійковому режимі

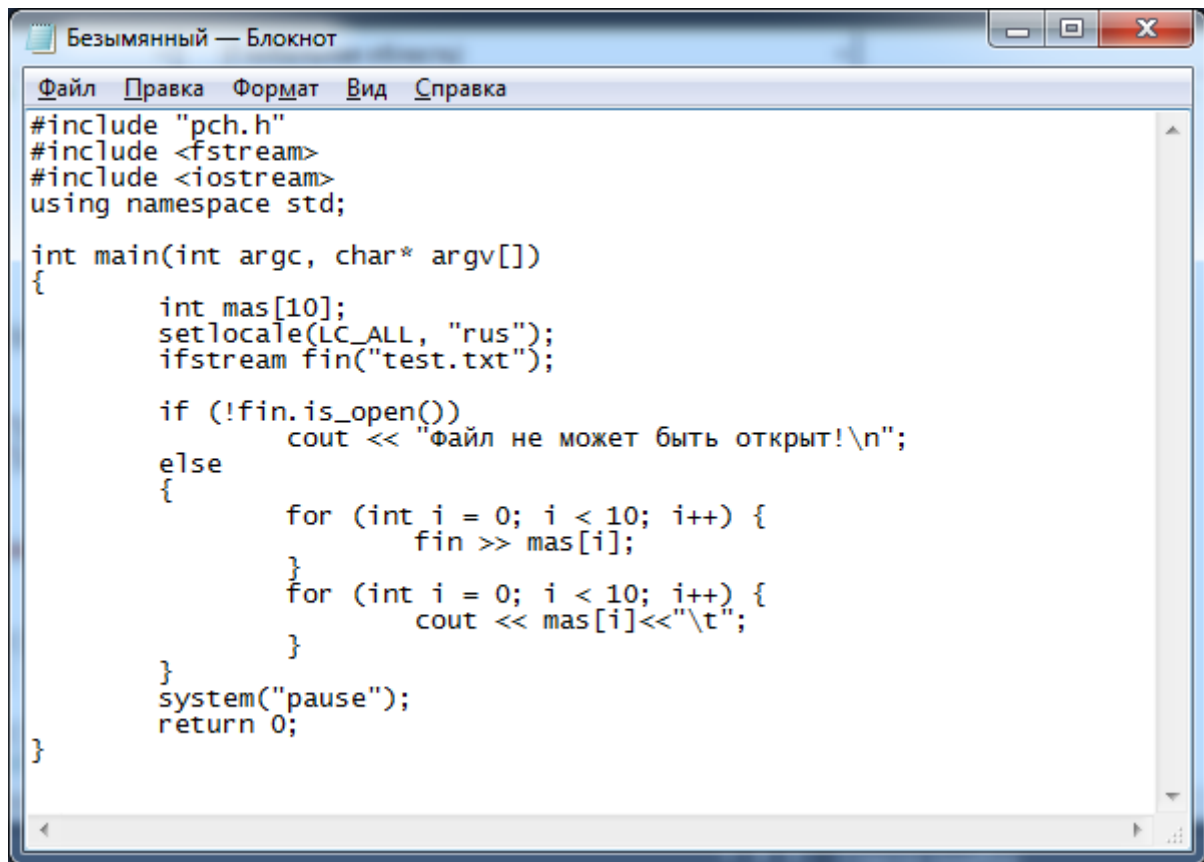
Режими відкриття файлів можливо встановлювати безпосередньо при створенні об'єкту або при виклику функції `open()`.

- 1 `ofstream fout("test.txt", ios_base::app);`
- 2 `fout.open("test.txt", ios_base::app);`

Режимів може бути декілька, вони поєднуються через логічне **ИЛИ**.

Питання: Які режими при відкритті файлу вже встановлені за замовчуванням?

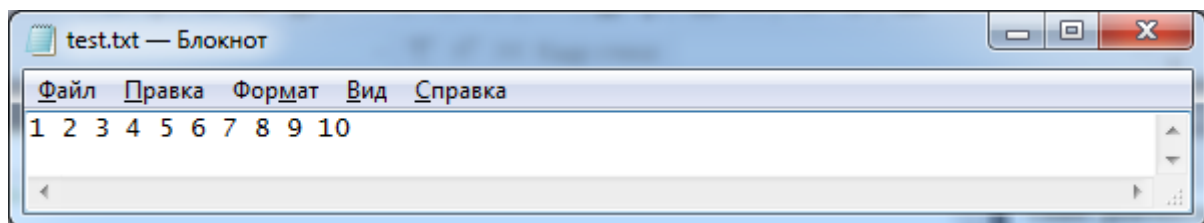
Нижче наведений приклад програми, де проходить зчитування елементів масиву файлу:



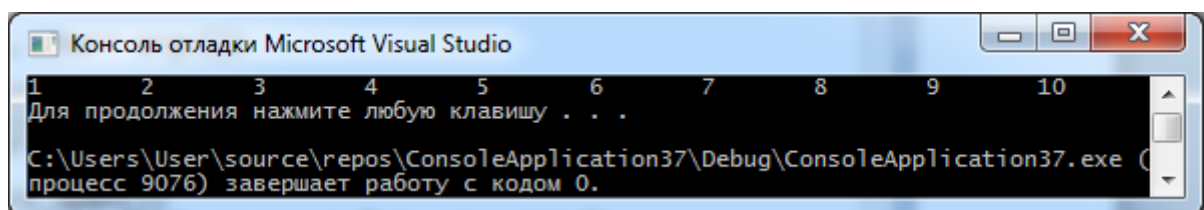
```
#include "pch.h"
#include <fstream>
#include <iostream>
using namespace std;

int main(int argc, char* argv[])
{
    int mas[10];
    setlocale(LC_ALL, "rus");
    ifstream fin("test.txt");

    if (!fin.is_open())
        cout << "Файл не может быть открыт!\n";
    else
    {
        for (int i = 0; i < 10; i++) {
            fin >> mas[i];
        }
        for (int i = 0; i < 10; i++) {
            cout << mas[i] << "\t";
        }
    }
    system("pause");
    return 0;
}
```



```
1 2 3 4 5 6 7 8 9 10
```



```
1      2      3      4      5      6      7      8      9      10
Для продолжения нажмите любую клавишу . . .
C:\Users\User\source\repos\ConsoleApplication37\Debug\ConsoleApplication37.exe (
процесс 9076) завершает работу с кодом 0.
```

Але що буде, коли елементів недостатньо?

З цим необхідно розібратись самостійно. На практиці продовжимо.