**Лекція 27 Введення- виведення даних у мовах С та С++ Файли**.

**Ввід/вивід**, **введення-виведення** ([англ.](https://uk.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D1%96%D0%B9%D1%81%D1%8C%D0%BA%D0%B0_%D0%BC%D0%BE%D0%B2%D0%B0" \o "Англійська мова) *I/O, input/output*) в інформатиці — взаємодія між обробником інформації (наприклад, комп'ютер) і зовнішнім світом, який може представляти як людина, так і будь-яка інша система обробки інформації. Введення, ввід — сигнал або дані, отримані системою, а вивід — сигнал або дані, надіслані нею (або з неї). Термін також може використовуватися як позначення (або доповнення до позначення) певної дії: «виконувати введення / виведення» означає виконання операцій введення або виведення. Пристрої введення-виведення використовуються людиною (або іншою системою) для взаємодії з комп'ютером. Наприклад, клавіатури та миші — спеціально розроблені комп'ютерні пристрої введення, а монітори та принтери — комп'ютерні пристрої виводу. Пристрої для взаємодії між комп'ютерами, як модеми та мережеві карти, зазвичай служать пристроями введення і виведення одночасно.

Варто відзначити, що призначення пристрою як пристрою введення або виведення залежить від перспективи. Миші та клавіатури приймають фізичну дію, здійснювану людиною-користувачем (до речі, щодо нього це будуть дії з виведення інформації), і перетворює його в сигнали, зрозумілі комп'ютеру. Виведення інформації з цих пристроїв є введенням її в комп'ютер. Аналогічно, принтери та монітори отримують на вході сигнали, які виводить комп'ютер. Потім вони перетворять ці сигнали в такий вигляд, який людина зможе побачити чи прочитати. Для людей-користувачів процес читання або перегляду подібних варіантів представлення інформації є введенням або отриманням інформації.

У комп'ютерній архітектурі об'єднання процесора та основної пам'яті (тобто пам'яті, з якої процесор може читати та записувати в неї прямо за допомогою особливих інструкцій) становить «мозок» комп'ютера, і з цієї точки зору, будь-який обмін інформацією з цим об'єднанням, наприклад, з дисковим накопичувачем, має на увазі введення-виведення. Процесор і його супутні електронні кола реалізують введення-виведення з розподілом пам'яті, використовувані в низькорівневому програмуванні при реалізації драйверів пристроїв.

Високорівнева операційна система і програмне забезпечення використовують інші, більш абстрактні концепції та примітиви введення-виведення. Наприклад, більшість операційних систем реалізують прикладні програми через концепцію файлів. Мови програмування Сі та C++, а також операційні системи сімейства Unix, традиційно абстрагують файли та пристрої у вигляді потоків даних, з яких можна читати та в які можна записувати, або і те й інше разом. Стандартна бібліотека мови Сі реалізує функції для роботи з потоками для введення і виведення даних.

Інтерфейс вводу-виводу вимагає керування процесором кожного пристрою. [Інтерфейс](https://uk.wikipedia.org/wiki/%D0%86%D0%BD%D1%82%D0%B5%D1%80%D1%84%D0%B5%D0%B9%D1%81) повинен мати відповідну логіку для інтерпретації адреси пристрою, що генерується процесором.

Встановлення контакту повинно бути реалізовано інтерфейсом за допомогою відповідних команд типу (ЗАЙНЯТИЙ, ГОТОВИЙ, ЧЕКАЮ), щоб процесор міг взаємодіяти з пристроєм вводу-виводу через інтерфейс.

Якщо існує необхідність передачі розрізнених форматів даних, то інтерфейс повинен вміти конвертувати послідовні (впорядковані) дані в паралельну форму і навпаки.

Повинна бути можливість для генерації [переривань](https://uk.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D0%B5%D1%80%D0%B8%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F) і відповідних типів чисел для подальшої обробки процесором (за потреби).

Комп'ютер, що використовує введення-виведення з розподілом пам'яті, звертається до [апаратного забезпечення](https://uk.wikipedia.org/wiki/%D0%90%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%B5_%D0%B7%D0%B0%D0%B1%D0%B5%D0%B7%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%BD%D1%8F) за допомогою читання і запису в певні [осередки пам'яті](https://uk.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D1%96%D1%80%D0%BA%D0%B0_%D0%BF%D0%B0%D0%BC%27%D1%8F%D1%82%D1%96), використовуючи ті ж самі інструкції мови [асемблера](https://uk.wikipedia.org/wiki/%D0%90%D1%81%D0%B5%D0%BC%D0%B1%D0%BB%D0%B5%D1%80), які комп'ютер зазвичай використовує при зверненні до пам'яті.

Вирішення навіть найпростішої задачі на комп’ютері не обходиться без операцій введення-виведення інформації. Введення даних - це передача інформації ззовні в ОП (із зовнішнього носія); виведення даних - зворотний процес, коли дані після обробки передаються з ОП на зовнішній носій. Зовнішнім носієм може служити дисплей, друкований пристрій, гнучкий або жорсткий диск тощо. Передача даних програмі та виведення результатів програми є необхідним елементом програми.

Програми можуть отримувати вхідні дані декількома способами: із стандартного вхідного файлу, пов'язаного з клавіатурою (наприклад, Input - в Pascal), або із дискового файлу. Аналогічний поділ існує і при виведенні даних: результати роботи програми за замовчуванням направляються на екран дисплея (у стандартний вихідний файл – наприклад, Оutput - в Pascal), але можна перенаправляти їх і на інший носій інформації, наприклад, на дисковий файл, на контролер будь-якого периферійного пристроюкомп'ютера тощо.

У сучасних мовах програмування основним поняттям введення та виведення даних є *потік* — послідовність символів або інших даних. У програмі потік зображує фізичний файл на зовнішньому носії даних (диску, клавіатурі або екрані монітора), тобто фізичний файл «видно в програмі» як потік даних. Операції обміну даних з файлом представлено в програмі як операції добування даних з потоку або дописування їх до нього. Cтандартним файлам введення та виведення у таких мовах відповідають стандартний потік введення (stdin - в С, cin - в C++) та стандартний потік виведення (stdout - в С, cout - в C++).

Алгоритмічні мови програмування використовують концепцію поелементного введення-виведення даних. Зазвичай, введення інформації з клавіатури супроводжується "ехо-сигналом" - на екрані дисплея з'являється зображення символів, що вводяться. В деяких системах програмування існують засоби, що дозволяють відключити таке ехо. Виведення даних організується, починаючи з позиції розміщення курсору. При цьому пробіли між даними, що виводяться, автоматично не вставляються - їх необхідно враховувати самим.

За допомогою відповідних засобів можна вводити або виводити дані тільки певних типів.

Організація введення та виведення даних у різних мовах програмування виконується по різному: або відповідними операторами, або за допомогою стандартних підпрограм. Так у мові С/С++ немає вбудованих засобів введення/виведення даних. Для організації відповідних дій використовуються стандартні бібліотечні функції.

Бібліотеки С/С++ підтримують два основних способи введення/виведення: потокове введення/виведення (заголовний файл fstream) та форматоване введення/виведення за допомогою функцій (заголовний файл stdio.h).

Потокове введення / виведення. Потокове виведення виконується за допомогою вихідного потоку cout та операції вставки у вихідний потік ”<<”. Стандартна операція ”<<” для операцій виведення перевизначається, тобто для неї визначаються нові можливості. Потокове введення аналогічне виведенню, але використовує перевизначену операцію вставки з вхідного потоку ”>>” для потоку введення cin.

Виконуючи операцію введення з клавіатури, комп’ютер тимчасово зупиняється і очікує на введення значення для змінної. У відповідь слід на клавіатурі набрати деяку послідовність символів, що зображує значення (ці символи з’являться на екрані). Уведені символи запам’ятовуються у буфері та передаються функціям введення тільки після натиснення клавіші Enter.

*Буфер* — це область пам’яті для тимчасового зберігання даних. Максимальний обсяг буфера становить 128 символів (байтів). Завдяки наявності буфера можливе редагування даних під час їх введення.

За стандартних налаштувань запис числа сприймається як десятковий. Якщо натиснути Enter, не набравши нічого, окрім пропусків, то комп’ютер і надалі чекатиме. В інструкції введення можна записати кілька імен змінних - кожне після «свого» знака ”>>”. При виконанні такої інструкції треба набрати на клавіатурі відповідну кількість вхідних констант, відокремивши їх одним або кількома порожніми символами.

Операція «помістити в потік» (”<<”)має асоціативність зліва направо:

(((cout<<"47+ 53 =")<<(47+53))<<endl).

Тобто, багаторазові операції ”<<” виконуються у тій послідовності, у якій вони записані. Такий спосіб об’єднання операцій ”<<” можливий, оскільки дана операція повертає посилання на об’єкт свого лівого операнда (на об’єкт cout). Отже, саме лівий вираз в круглих дужках

(cout << "47 + 53 =")

виводить заданий рядок символів і повертає посилання на cout*.* Тому середній вираз в круглих дужках має виконуватися, тобто виводити ціле значення 100 і повертає посилання на cout*,* наприклад: (cout<<(47+53)). Потім виконується самий правий вираз у круглих дужках, наприклад: cout << endl*,* який переводить рядок, скидає cout і повертає посилання на cout. Аналогічно, багато­разовою є і операції ”>>”.

Для керування потоковим введенням/виведенням можуть використовуватися маніпулятори потоків, які дозволяють міняти режими виконання операцій введення-виведення. Наприклад, маніпулятор hex задає форматування із шістнадцятковим перетворенням, endl - вставку символу нового рядка і очищення потоку.

Стандартні засоби потокового введення/виведення описані у бібліотеці iostream (iostream.h), маніпулятори потоків - у бібліотеці iomanip.h. Щоб мати змогу їх використати у програмі, відповідні бібліотеки слід підключити до програми за допомогою директиви #include. Наприклад,

#include <iostream.h>

int main()

{ double r,s;

const float pi=3.14;

cout<< "\nr="; cin >> r;

s=pi\*r\*r;

cout<< "\nПлоща кола: " << s << endl;

system("pause");

return 0;

}

Форматоване введення / виведення.Здійснюєтьсяфункціями:

* printf - виводить аргументи у стандартний потік stdout у відповідності із заданим форматом;
* scanf - вводить дані із стандартного потоку вводу stdin у відповідності із заданим форматом, записуючи їх у змінні, адреси яких задаються аргументами.

Формат виклику функції printf:

printf("форматний\_рядок", [список\_аргументів]),

де список\_аргументів = аргумент {, аргумент}

форматний\_рядок = “”” ([літерал] | % [маркер][ширина][.точність] тип)

{( [літерал] | % [маркер][ширина][.точність] тип}”””

Форматний рядок завжди береться в подвійні лапки. Символ "%" є символом початку задання формату введення чергового аргумента. При описі формату виведення також використовуються:

маркер – специфікатор вирівнювання виведених знаків, пропусків, десяткових точок, вісімкових і шістнадцяткових префіксів (наприклад, вирівнювання результату перетворення по лівій межі, обов'язкове зображення знака числа тощо);

ширина– загальна ширина поля виведення;

точність – максимальне число цифр дробової частини числа, які будуть виведені після коми;

тип – специфікатор типу аргумента (наприклад, d – ціле десяткове число, f – дійсне число).

Функція printf переводить дані з внутрішнього коду в символьне представлення відповідно до форматного рядка і виводить отримані символи на екран. Форматний рядок може включати довільний текст, керуючі символи та специфікації перетворення даних. Список аргументів є необов'язковим параметром даної функції. Дану функцію можна використовувати для виводу будь-якої комбінації символів, цілих та дійсних чисел, тощо.

Формат виклику функції scanf:

scanf("форматний\_рядок", список\_адрес\_аргументів),

де список\_ адрес \_аргументів = &аргумент {, &аргумент}

форматний\_рядок = ([літерал] | %[ширина\_поля] тип) { ([літерал] | %[ширина\_поля] тип}

Аргументами функції scanf є не імена змінних, а їхні адреси (символ & означає взяття адреси). Функція scanf зчитує коди з клавіатури, перетворює їх у відповідності із специфікацією форматного рядка і передає програмі. Форматний рядок і список аргументів для функції scanf – обов'язкові параметри. Описи відповідних функцій введення / виведення містяться у заготовочному файли stdio.h.

Наприклад, програма обчислення площі кола:

#include <stdio.h>

void main()

{ double r,s;

const float pi=3.14;

printf("\n r=");

scanf("%lf",&r);

s=pi\*r\*r;

printf("\nПлоща кола: %10.3f",s);

system("pause");

return 0;

}

### C

Для виведення повідомлень в С використовують стандартну бібліотеку введення/виведення [*stdio*](http://www.cplusplus.com/reference/cstdio/) та функцію форматного виведення [printf](http://www.cplusplus.com/reference/cstdio/printf/).

Для включення цією бібліотеки до програми необхідно використати директиву препроцесора #include.

#include <stdio.h>

Використання printf подібне до використання System.out.printf в Java.

Виведення

|  |
| --- |
| **Код** |
| printf (“Characters: %c %c \n”, ‘a’, 65); printf (“Decimals: %d %ld\n”, 1977, 650000L); printf (“Preceding with blanks: %10d \n”, 1977); printf (“Preceding with zeros: %010d \n”, 1977); printf (“Some different radices: %d %x %o %#x %#o \n”, 100, 100, 100, 100, 100); printf (“floats: %4.2f %+.0e %E \n”, 3.1416, 3.1416, 3.1416); printf (“Width trick: %\*d \n”, 5, 10); printf (“%s \n”, “A string”); |
| **Виведення** |
| haracters: a A Decimals: 1977 650000 Preceding with blanks:       1977 Preceding with zeros: 0000001977 Some different radices: 100 64 144 0x64 0144 floats: 3.14 +3e+000 3.141600E+000 Width trick: 10 A string |

### Заголовок <stdio.h>

### Введення (вв) — це процес передачі даних під час виконання програми з зовнішніх пристроїв в оперативну пам’ять, а виведення (вив) — зворотній процес пересилки даних з оперативної пам’яті назовні.

### Вв/вив даних реалізовано у мові С не спеціальними операторами, як у деяких інших мовах, а функціями.

### Декларації цих функцій містяться у стандартній бібліотеці, і їх треба підключати за допомогою заголовного файлу <stdio.h>. Вміст stdio.h залежить від реалізації. Тіла функцій, тобто програмний код, як правило, не містяться в <stdio.h>, вони поставляються у прекомпільованому вигляді, підключаються під час лінкування С-програми.

### Згідно стандарту, заголовок <stdio.h>:

### - визначає кілька макросів (зокрема, NULL, який розширюється до визначеного реалізацією нульового вказівника-константи; \_IOFBF, \_IOLBF, \_IONBF, які розширюються до цілих констант, придатних для використання в якості третього аргументу функції setvbuf; EOF, який розширюється до цілої константи типу int та від'ємним значенням, що повертається декількома функціями при виявленні кінця файлу; stderr, stdin, stdout, які є виразами типу вказівник на тип FILE і які вказують на FILE-об'єкти, асоційовані відповідно зі стандартними потоками помилок, введення та виведення;

### - декларує три типи (один з них FILE);

### - декларує багато функцій для виконання введення та виведення.

### 

### В мові С використано поняття потік (stream) та файл (file) для програмування процесів обміну даними (рис. 1).

### Файл — це або розміщений на пристрої довільний блок інформації з унікальним іменем, або сам фізичний пристрій (термінал, клавіатура, диск, флешка, мережа тощо), ототожнюваний з файлом.

### Згідно стандарту, потік даних - абстрактне поняття для узагальнення процесу передачі даних для різних типів пристроїв та файлів.

### Якими б не були пристрої чи файли, дані відображаються або у текстові потоки, або у двійкові потоки (рис. 1).

### 

### Рисунок 1 – Організація обміну даними в мові С

### Потоки. Файли

### В мові С використано поняття потік (stream) та файл (file) для програмування процесів обміну даними (рис. 1).

### Файл — це або розміщений на пристрої довільний блок інформації з унікальним іменем, або сам фізичний пристрій (термінал, клавіатура, диск, флешка, мережа тощо), ототожнюваний з файлом.

### Згідно стандарту, потік даних - абстрактне поняття для узагальнення процесу передачі даних для різних типів пристроїв та файлів.

### Якими б не були пристрої чи файли, дані відображаються або у текстові потоки, або у двійкові потоки (рис. 1).

### Отже, програмісту надається наступна “формула” для оволодіння інструментом обміну даними в С-програмах:

### ФАЙЛ -------> ПОТІК -------> ФУНКЦІЇ ВВ/ВИВ.

### Для роботи з потоками програмісту треба ознайомитись з похідним типом даних FILE.

### Згідно стандарту, тип FILE є типом даних для представлення об’єктів – потоків.

### Об’єкт типу FILE містить інформацію для управління потоком, зокрема:

### - індикатор позиції у файлі,

### - вказівник на асоційований буфер (якщо такий є),

### - індикатор помилки, яка відбулася при вв/вив,

### - індикатор кінця файлу, в який записується, чи був досягнутий кінець файлу.

### Якщо потік не буферизований (unbuffered), символи мають передатись (ввестися / вивестися) якнайшвидше.

### Якщо потік буферизований, символи можуть накопичуватись у буфері і передаватись як блок.

### Якщо потік повністю буферизований (fully buffered), символи передадуться при наповненні буфера. Якщо потік буферизується рядком (line buffered),символи передадуться блоком, коли зустрінеться символ кінця рядка.

### Ось для прикладу фрагмент файлу stdio.h, який показує декларацію типу даних FILE. З декларації видно, що тип даних FILE – це похідний тип – структура з тегом \_iobuf. Значення елементів структури використовуються середовищем виконання для управління потоком.

### …

### typedef struct \_iobuf

### {

### char\* \_ptr;

### int \_cnt;

### char\* \_base;

### int \_flag;

### int \_file;

### int \_charbuf;

### int \_bufsiz;

### char\* \_tmpfname;

### } FILE;

### …

### Тип FILE запропоновано для зручності, для підвищення рівня програмування вв/вив в С. Програмісту аж ніяк не потрібно вникати в деталі елементів структури, тим більше, що вміст stdio.h залежить від реалізації.

### Коли запускається функція main, у неї вже є три попередньо визначені стандартні потоки, відкриті та доступні для використання. Ці потоки декларуються у файлі заголовка stdio.h:

### FILE \* stdin - стандартний потік введення, наприклад, з клавіатури

### FILE \* stdout - стандартний потік виведення, наприклад, на монітор

### FILE \* stderr - стандартний потік для виведення повідомлень про помилки

### та діагностики при виконанні програми

### Отже, програміст має виконати такі кроки для організації роботи програми з файлами:

### 1. Визначитися з іменами потрібних файлів. Ім’я файлу - це рядок символів.

### 2. Написати декларації вказівників на асоційовані з файлами потоки.

### Вказівник потоку - це вказівник на об'єкт типу FILE; цей об'єкт описує асоційований з файлом потік. Для стандартних потоків введення з клавіатури та виведення на екран цей пункт пропускається, оскільки підтримується системою.

### 3. Відкрити файли за допомогою відповідних функцій. Отримати від цих функцій значення вказівників на асоційовані потоки.

### 4. Застосувати вказівники при викликах функцій вв/вив, обраних для організації подальшого “руху” даних.

### Табл. 2 містить опис бібліотечних функцій, які здійснюють наступні операції над файлами:

### - видалення файлу,

### - перейменування файлу,

### - створення тимчасового файлу,

### - генерація унікального тимчасового імені файлу.

### Таблиця 2 – Функції операцій над файлами

### 

### 

### 

### 

### Табл. 3 містить опис бібліотечних функцій доступу до файлів:

### - відкриття файлу,

### - закриття файлу,

### - очищення буферу,

### - буферізація потоку.

### Табл. 3 містить опис бібліотечних функцій

### 

### 

### Табл. 4 містить опис бібліотечних функцій форматованого вв/вив.

### Форматоване виведення перетворює внутрішнє двійкове представлення даних у символи, які записуються у файл.

### Форматоване введення зчитує символи з файлу і перетворює їх у внутрішню форму.

### Форматоване вв/вив дуже портативне: файли з форматованими даними легко переносити на різні комп'ютери з різними операційними системами, звісно, якщо всі вони використовують стандартний набір символів (ASCII). Форматовані файли є читабельними для людей і можуть бути набрані з клавіатури або відредаговані за допомогою текстового редактора.

### Проте для роботи з форматованими даними потрібно більше ресурсів часу та пам’яті, аніж з неформатованими.

### Таблиця 4 - Функції форматованого введення та виведення

### 

### 

### Таблиця 5 - Функції символьного введення та виведення

### 

### 

### C++

В C++ залишається можливість використання бібліотеки stdio, однак частіше використовують потокове введення/виведення з бібліотеки [*iostream*](http://www.cplusplus.com/reference/iostream/)

#include <iostream>

Дана бібліотека містить простір імен std який містить оператор cout (console out). Даний оператор містить наступний синтаксис:

std::cout<<operand1<<operand2<<operand3<<…;

Перед кожний елементом який необхідно вивести на екран записуються через << (дві кутні дужки вліво/два знаку менше)

Наприклад:

cout<<“Slava “<<31;

cout<<x<<y<<z;

За замовчуванням пробіли між елементами при виведенні НЕ ставляться тому за необхідності їх треба поставити самому:

cout<<x<<” “<<y<<” “<<z;

### Особливості виведення:

Такі мови як Java, C/C++ підтримують спеціальні керуючі послідовності символів у рядках. Які можна використовувати при виведенні. Усі такі послідовності починаються із символа “\” (backslash)

* \n – новий рядок
* \t – табуляція

Оскільки деякі деякі символи в мові програмування використовуються для позначення тих чи інших елементів, і розпізнаються як частина мови програмування, для їх виведення використовують додаткові керуючу послідовності

* \\ – символ \
* \” – символ “
* \’ – символ ‘
* \% – символ %
* \? – символ ?

Приклад:

Printf(“Some\tText\nSpecial symbols:\n\\ – backslash\n\% – persecnt\n\” – quotes\n\’ – apostrophe.”):

Виведення:  
Some     Text

Spescial symbols:

\ – backslash

% – persecnt

” – quotes

‘ – apostrophe.

## Введення

Якщо дані, які треба опрацювати завчасно не відомі, або можуть мати різне значення, для подальшої роботи з ними необхідно це значення дізнатись. Програма може отримувати дані з будь-яких джерел, одне з них – клавіатура

Для введення даних з клавіатури у консольному режимі роботи використовують стандартні оператори введення з клавіатури.

**Поняття файлу**

Під час розв’язання задач на комп’ютері часто виникає необхідність у використанні даних, які записані на зовнішніх носіях інформації (дисках) і оформлені у вигляді файлів даних. Незалежно від того, які дані містять файли (числа, символи, рядки, масиви, структури тощо), в мові С++ вони трактуються як потоки даних **(stream),** які являють собою послідовність байтів, що зчитуються або записуються.

За замовчуванням у кожній програмі С++ можна користуватися такими стандартними потоками: стандартного введення **(сіn),** стандартного виведення **(cout)** та виведення помилок **(сеrr).** Щоб користуватися файлами, потоки повинні бути створені і закріплені за цими файлами.

**Використання файлів даних у програмі передбачає виконання таких операцій**:

* створення потоку обміну даними між файлом і пам’яттю комп’ютера;
* зв’язування цього потоку з конкретним ім’ям файлу на диску і відкриття файлу;
* запис даних у файл або читання їх з файлу;
* закриття файлу.

Для реалізації цих операцій існують спеціальні класи (структури даних разом з функціями обробки цих даних), які містять конструктори створення необхідних потоків:

* **ifstream** — для створення потоку читання даних;
* **ofstream** — для створення потоку запису даних у файл;
* **fstream** — використовується як для запису даних у файл, так і їх читання.

**Конструктори для роботи з файлами**

**Конструктори** — це спеціальні функції, які мають таке саме ім’я, що й ім’я класу. Вони записуються як з параметрами, так і без параметрів. Конструктори з параметрами одночасно створюють відповідний потік, зв’язують його з файлом на диску, відкривають файл для роботи і мають такі форми запису:

**ofstream іп (“іф”, ios::out); або ofstream (“іф”);**

**ifstream іп (“іф”, ios::in); або ifstream (” “);**

**fstream іп (“іф”, ios::in | ios::out);**

де **іп** — ім’я потоку, який створюється для роботи з файлом;

**іф** — константа або змінна типу **char[ ],** її значення — ім’я файлу на диску.

Перший з конструкторів використовується для запису даних у файл, другий — для читання даних з файлу, а третій — як для запису, так і для читання даних, наприклад:

**ofstream fout( "myfile.dat", ios::out);**

Цей запис створює потік з ім’ям **fout**, зв’язує його з файлом на диску, який має ім’я **myfile.dat** і відкриває цей файл для запису даних. Файл **myfile.dat** буде створено у тому ж каталозі, що і програма. Якщо треба створити файл у другому місці, то для запису його імені треба вказати шлях, наприклад, **a:\\pvp\\myfile.dat**. Тепер цей файл буде записано на диску **а:** в каталозі **pvp**.

Зверніть увагу на те, що для запису шляху треба використовувати подвійні зворотні косі риски.

Можна також для роботи з файлами застосувати конструктори без параметрів:

**ofstream іп;**

**ifstream іп;**

**fstream іп;**

де **іп** — ім’я відповідного потоку, тоді для зв’язування потоку з ім’ям файлу на диску і відкриття його для роботи треба додатково використовувати функцію-член відповідного класу, тобто:

**іn.ореn(“іф”, ios :: режим | ios :: режим);**

(в цьому запису  **|** означає "або", тобто можливість завдання декількох режимів).

Наприклад, відкриття файлу для запису до нього даних матиме вигляд:

**ofstream fout;**

**fout.open ( "a:\\pvp\\my file.dat", ios::out);**

Конструктори як з параметрами, так і без них, виконують однакову роботу, тому яким з них надати перевагу — справа користувача.

***Приклад***  Створити файл на диску і записати до нього масив чисел. Прочитати цей файл і вивести його компоненти на екран.

// Створення файлу та запис до нього масиву

**#include <iostream>**

**#include <fstream>**

**#include <conio.h>**

**#include <Windows.h>**

**using namespace std;**

**int main()**

**{ int i, mas[5];**

**system("color F0");**

**// запис елементів масиву до файлу:**

**ofstream fout("massiv.txt"); /\* створення потоку fout та відкриття файлу з іменем massiv. txt для запису \*/**

**if (! fout) cout <<"Cannot open file\n";**

**for (i = 0; i <5; i++)**

**{**

**cout << " Enter "<< i << " element\n";**

**cin >> mas[i]; // введення елементу масиву з клавіатури**

**fout << mas[i] << " "; //запис елементу до файлу**

**}**

**fout.close();**

**//---------- читання компонентів файлу та виведення на екран**

**ifstream fin("massiv.txt"); /\* створення потоку fin для читання файлу\*/**

**if (!fin) cout << " Cannot open file fo reading\n";**

**else {**

**cout <<"REZULTAT \n";**

**for (i = 0; i <5; i++)**

**{**

**fin >> mas[i];**

**// читання поточного елементу масиву з файлу**

**cout << "mas[" << i << "]=" << mas[i] << " ";**

**}**

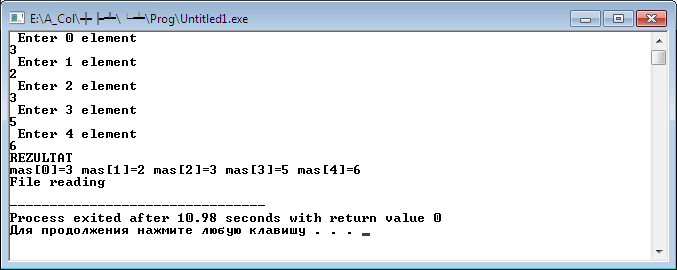
**cout << "\nFile reading\n";};**

**fin.close();**

**return 0;**

**}**

Результат розв’язання прикладу має вигляд:



*Для самостійного вивчення (10 годин)*: Вивчення лекційного матеріалу та додаткових джерел. Розгляд запитань і виконання завдань для самостійної роботи, запропонованих на лекції.

*Література*

1. Ковалюк Т. В. Алгоритмізація та програмування: Підручник. — Львів: «Магнолія 2006», 2013. — 400 с., ил.
2. Єжова Л. Ф. Алгоритмізація і програмування процедур обробки інформації: Навч.-метод. посіб. для самост. вивч. дисц. К.: КНЕУ, 2000.
3. Вступ до програмування мовою С++. Організація обчислень: навч. посіб. / Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставовський. – К. : Видавничо-поліграфічний центр "Київський університет", 2012. – 175 с. с.: іл. ISBN (укр.) . URL: <http://csc.knu.ua/uk/library/books/belov-24.pdf>
4. Джейс Либерти Освой самостоятельно С++ за 21 день: 3-е изд. пер. с англ.: Уч. пос. – М.: Издательский дом „Вильямс”, 2001. – 816 с.: ил..

*Контрольні запитання для самоперевірки*.

1. Які змінні називаються локальними?
2. Розкажіть про порядок дій, які виконуються під час виклику функції.
3. Які параметри називаються формальними, а які фактичними?
4. Чи можна у викликаємій функції змінювати параметри при **виклику функції з передачею значень? Чи вплине така зміна значень параметрів на дії в головній функції, якщо ці параметри потім там використовуються?**
5. **Які правила встановленні для формування списку параметрів, коли** передача даних виконується за замовчуванням?

*Контрольні запитання для надання письмових відповідей*.

1. .Що потрібно зробити, щоб зберегти значення в середині функції між її викликами? Наведіть приклад у вигляді фрагменту коду.
2. Розгляньте останній приклад , виберіть всі мовні конструкції щодо оброблення файлів, які ми детально не розглядали в лекції.