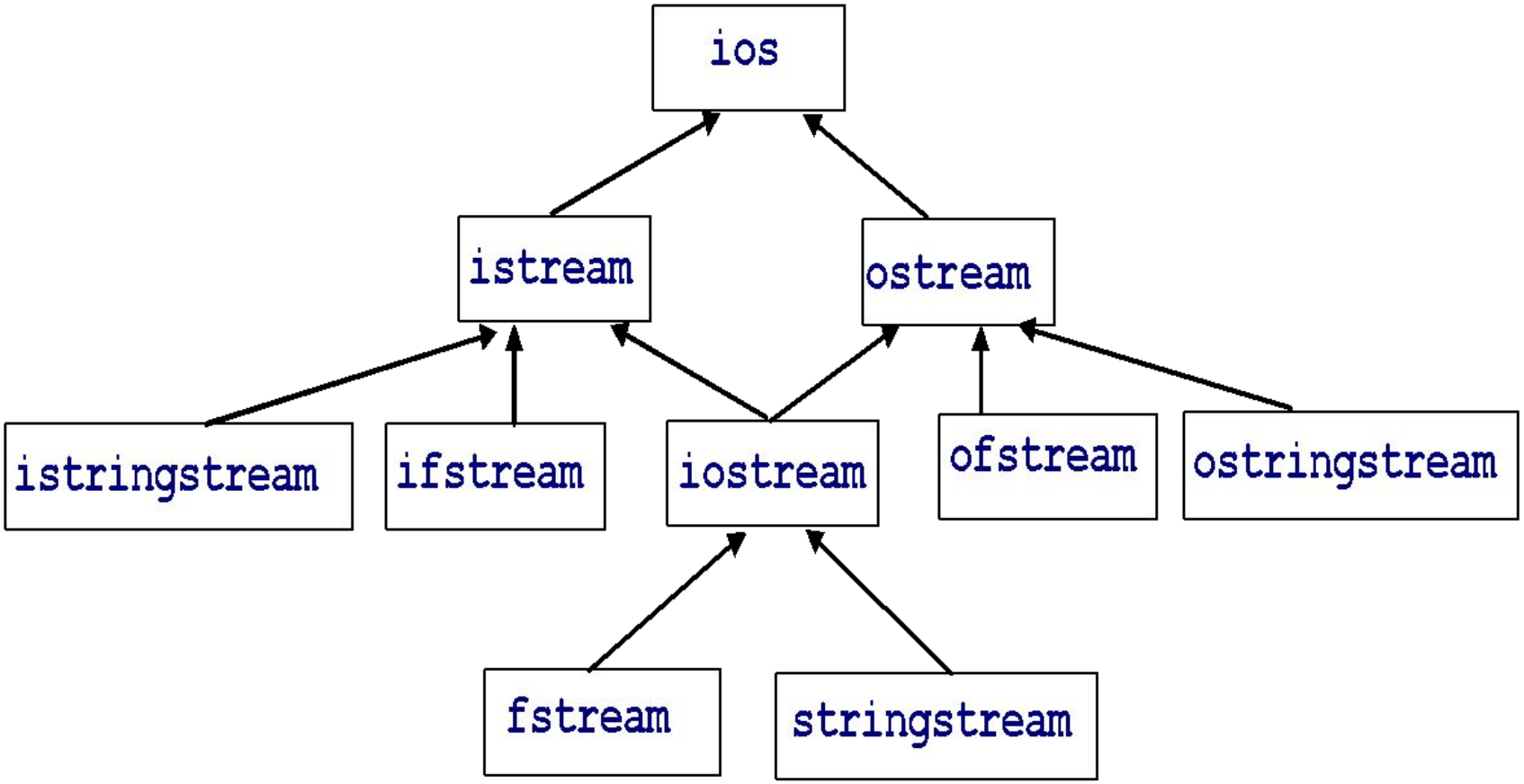
Огляд введення та виведення в C++

Заголовочний файл **`iostream`** містить набір класів, що забезпечують так зване потокове введення та виведення інформації в стилі мови C++. З точки зору програми C++, введення та виведення являє собою потік байтів. При введенні програма вибирає байти з потоку вводу, при виведенні – відправляє байти в потік. Якщо програма орієнтована на роботу з текстом (або навіть з числовою інформацією в текстовому вигляді), кожний байт є представленням чергового символу. В загальному випадку байти потоку є двійковим представленням даних довільного типу.

Байти потоку введення можуть поступати з клавіатури або з будь-якого зовнішнього пристрою пам'яті. Так само байти потоку виведення можуть бути відправлені на екран, принтер, зовнішній пристрій. Тобто потік діє як посередник між відправною точкою та точкою призначення потоку. Це дозволяє С++-програмі обробляти потоки незалежно від того, звідки вони поступили або куди направляються. Образно потік можна уявляти як трубу, по якій спрямовуються байти інформації. Отже, для роботи з потоком його треба приєднати до програми та зв'язати з файлом, що містить інформацію. Більш ефективно введення та виведення інформації здійснюється з допомогою буфера, який відіграє роль резервуара для накопичення інформації щоб зменшити кількість звертань до зовнішніх пристроїв пам'яті.

Ієрархія класів потоків мови C++.



Для введення/виведення у файл/з файлу використовують потоки, які можуть бути пов'язані із файлом на диску

Для їх використання потрібно підключити заголовочний файл **<fstream>**. Існує 3 різновиди файлових потоків: **fstream**, **ifstream** (для введення інформації), **ofstream** (для введення інформації). Змінити поведінку потоку, передбачену умовчанням, можна з допомогою наступних керуючих констант:

ios::app – відкриття файлу для доповнення;

ios::binary – відкриття двійкового, а не текстового файлу;

ios::in – відкриття файлу для читання;

ios::out – відкриття файлу для запису;

ios::trunc – видалення змісту файлу при відкритті.

Вказані режими відкриття файлу можна комбінувати операцією побітового додавання АБО - |

Деякі приклади відкриття файлу.

`fstream fs("f1.txt");`	**Відкриття файлу для читання та запису**
`ifstream ifs("f2.txt");`	**Відкриття файлу для читання**
`ofstream ofs("f3.txt");`	**Відкриття файлу та запису**
`fstream fs("f1.txt", ios::in | ios::out | ios::trunk);`	**Відкриття файлу для читання та запису з видаленням змісту**
`ifstream ifs("f2.txt", ios::in | ios::binary);`	**Відкриття двійкового файлу для читання**
`ofstream ofs;`	**Створюємо потік, не пов'язаний із файлом**
`ofs.open("f3.txt");`	**Відкриваємо файл для запису**

Клас `ifstream`, який реалізує операції введення із файлового потоку – більш детально.

Один із його конструкторів цього класу має вигляд:

```
ifstream :: ifstream (char* pFileName,
   int mode = ios::in);
```

Перший аргумент – ім’я файлу, що відкривається для введення. Другий задає режими відкриття файлу. Деякі можливі значення параметру **`mode`** представлені у таблиці нижче.

Клас `ofstream` , який реалізує операції виведення у файловий потік – більш детально.

Один із його конструкторів цього класу має вигляд:

```
ofstream :: ofstream (char* pFileName, int mode = ios::out);
```

Перший аргумент – ім’я файлу, що відкривається для виводу. Другий задає режими відкриття файлу. Деякі можливі значення параметру **`mode`** представлені у таблиці нижче.

Значення параметру **mode**

Прапорець	Призначення
ios::ate	Перемістити вказівник у кінець файлу після його відкриття
ios::app	Якщо файл існує – дозапис у нього
ios::in	Відкрити файл для введення (замовчування для **istream**)
ios::out	Відкрити файл для виведення (замовчування для **ostream**)
ios::nocreate	Якщо файл не існує – повернути помилку
ios::noreplace	Якщо файл існує – повернути помилку
ios::binary	Відкрити файл як двійковий (бінарний) (за замовчуванням як текстовий)

Функція-член **`bad()`** повертає 1, якщо при створенні екземпляру виникла помилка:

```
if (bin.bad())
{cerr << "Opening file error";}
```

Функція-член **`clear()`** скидає прапорець помилки (інакше спроби виведення блокуються):

```
bin.clear();
```

Потоки автоматично закриваються при завершенні програми. Однак при необхідності можна закрит потік функцією **`close()`**, а потім знову відкрити його, зв'язавши з іншим файлом.

Для перевірки правильності відкриття файлу можна використати і інші способи:

метод **is_open ()** повертає 0 (false), метод якщо **open ()** не спрацював успішно. Раніше більш поширеною була практика використання методу good **()**, який повертає 1 (true) в разі успішного відкриття файлу. Наприклад,

```
if (bin.is_open ())
{
// працюємо з файлом
}
```

або

```
if (bin.good ())
{
// працюємо з файлом
}
```

Для операцій з потоками використовуються перевантажені операції **<<** та **>>**, а також методи класу **ios** та похідних від нього класів. Розглянемо деякі з них.

Якщо **ch** змінна типу **char**, то вона може бути прочитана із файлу одним із методів:

```
ch = <екземпляр потоку>.get ();
```

або

```
<екземпляр потоку>.get (ch);
```

Якщо **ch** масив розміру **size** типу **char**, то він може бути заповнений символами із файлу тим же методом:

```
<екземпляр потоку>.get (ch, size, '\n');
```

або

```
<екземпляр потоку>.getline (ch, size, '\n');
```

(Останній параметр – не обов'язковий, він має умовчання)

Ще один спосіб одержати черговий елемент з файлу – використання операції **<<**.

Для запису у файл символьної змінної використовується метод `<екземпляр потоку>.put (ch);` або операція **>>**.

Для організації прямого доступу до файлу використовують функції `seekg/seekp` та `tellg/tellp`. Різниця між ними в тому, що функції з іменем, що закінчуються на символ `'g'`, використовуються для роботи із потоками введення, а функції, з іменем, що закінчуються на символ `'p'`, – для роботи з потоками виводу.

Функції `seekg/seekp` переміщують внутрішній вказівник файлу на задану позицію. Позиції відповідають байтам, нумерація починається з 0. Існує по 2 різновиди функцій – з одним параметром та з двома параметрами. Один цілий параметр задає абсолютную позицію у файлі. Два параметри задають зсув(ціле число) та точку відліку. Останній параметр може набувати таких значень:

- `ios::beg` – початок файлу;
- `ios::cur` – поточна позиція вказівника;
- `ios::end` – кінець файлу.

Функції `tellg/tellp` не мають параметрів. Вони повертають поточну позицію вказівника у файлі.

Для роботи з двійковими файлами використовуються функції `read` и `write`. В ролі параметрів функції одержують вказівник (типу `char*` для функції `read` та типу `const char*` для функції `write`), який задє адресу початку масиву для введення/виведення, та ціле число – кількість байтів для введення/виведення.

Прості операції з файлами введення.

Припустимо, що програма має читати інформацію із файлу. Для цього необхідно виконати наступні кроки:

1. Створити екземпляр класу **ifstream**, для керування потоком введення:

   ```
   ifstream ifile;
   ```

2. Поставити цей екземпляр у відповідність конкретному файлу:

   ```
   ifile.open ("D:\\in.txt");
   ```

(Кроки 1 та 2 можна замінити створенням екземпляру з допомогою конкретного конструктора:

```
ifstream ifile ("D:\\in.txt", ios::in);
```
)

3. Працювати із цим об'єктом так само, як із об'єктом **cin**.

Прості операції з файлами виведення.

Припустимо, що програма має виводити інформацію у файл. Для цього необхідно виконати наступні кроки:

1. Створити екземпляр класу **`ofstream`**, для керування потоком виведення:

   ```
   ofstream ofile;
   ```

2. Поставити цей екземпляр у відповідність конкретному файлу:

   ```
   ofile.open ("D:\\in.txt");
   ```

(Кроки 1 та 2 можна замінити створенням екземпляру з допомогою конкретного конструктора:

```
ofstream ofile ("D:\\in.txt", ios::out);
```
)

3. Працювати із цим об'єктом так само, як із об'єктом **`cout`**.