Лекція 13

Тема лекції: Потоки. Форматування виведення. Використання маніпуляторів потоку. Функції-члени класів IOS, ISTREAM, STREAM. Прапори форматування об'єктів класу IOS. Рядково-орієнтоване введення/виведення. Символічні константи режиму відкриття потоку. Перевантаження оператора виведення в потік

Потоки введення-виведення

В мові C++ як і в C, відсутні вбудовані оператори введення-виведення. У програмах на C та на C++ введення і виведення зазвичай здійснюються за допомогою функцій введення-виведення, що утримуються у бібліотеках. Стандарт мови C визначає набір функцій введення-виведення (*stdio.h*), які обов'язково повинні входити до складу стандартної бібліотеки C.

Прототипи стандартних функцій введення-виведення утримуються в заголовочному файлі *stdio.h*. Хоча всі стандартні функції C також доступні в C++, при програмуванні на C++ у багатьох випадках краще використовувати спеціальну стандартну бібліотеку потокового введення/виведення *iostream,* що містить великий набір класів і функцій для буферизованого і небуферизованого введення/виведення даних як для файлів, так і для пристроїв.

Що ж таке, по суті, потоки введення/виведення? У повній відповідності з логікою C++ потік визначається як деякий абстрактний тип даних, що представляє якусь послідовність елементів даних, спрямовану від джерела до споживача. Кількість елементів у потоці називають його довжиною, порядковий номер доступного в деякий момент елементу називають поточною позицією. Кожен потік має один із трьох можливих режимів доступу: тільки для читання, тільки для запису, для читання і запису. Бібліотека *iostream* містить дві паралельні родини класів. Перша веде свій початок від класу *streambuf*, а друга - від *ios*. Клас *streambuf* і два класи, похідних від нього - *filebuf* та *strstreambuf* забезпечують буферизацію потокового

введення/виведення і абстрагують звернення до фізичних пристроїв введення/виведення. Клас *ios* дає початок родині класів, призначених для реалізації форматованого і неформатованого введення/виведення на високому рівні з можливістю контролю і зміни стану потоку. *Ios* містить член *ios::bp* - покажчик на *streambuf*, за допомогою якого і спілкується з пристроями введення/виведення. Опис класу *ios* розташований у файлі *iostream.h*, включення якого до програми, що використовує введення/виведення, обов'язкове.

Серед класів, похідних від *ios* назвемо такі:

istream - введення зі стандартного пристрою введення. Містить перевантажений оператор форматованого введення ">>" та низку функцій неформатованого введення;

ostream - виведення на стандартний пристрій виведення. Містить перевантажений оператор форматованого виведення "<<" та низку функцій неформатованого виведення;

iostream - суміш класів *istream* і *ostream*;

iostream_withassign - *iostream* із перевантаженим оператором привласнення. Його об’єктами є стандартні об'єкти введення/виведення. У C++ це є об’єкти *cin*, *cout*, *cerr*, *clog*.

У заголовочному файлі *IOSTREAM.H* утримуються такі оголошення одного потоку введення і трьох потоків виведення:

- *in* - об'єкт потоку введення;
- *out* - об'єкт потоку виведення;
- *err* - об'єкт потоку виведення помилок;
- *log* - об'єкт буферизованого потоку виведення помилок.

Потокові класи виконують форматоване введення-виведення з вбудованою підтримкою обробки помилок. Потоки введення/виведення підтримують базові типи даних, подібні *char*, *short*, *int*, *long*, *float*, *double*, *long double*, *char** та *void** (значення - адреса).

Файлове введення-виведення. Розглянемо докладніше використання класу *ifstream* і *ofstream*:

ifstream - похідний від *istream* клас, пов'язує файл із прикладною програмою для виконання тільки введення.

ofstream - похідний від *ostream* клас, пов'язує файл із прикладною програмою для виконання тільки виведення.

Описи цих класів утримуються в заголовочному файлі *fstream.h.* Він автоматично підключає до програми і файл опису класів потоків *iostream.h.* Класи *ifstream* і *ofstream* є похідними від *istream* і *ostream* і успадковують операції *"<<"* та *">>"*.

Для відкриття файла для запису, використовується запис:

```
ofstream foo("ім'я файлу");
```

Для запису до файлу:

```
foo << "Hello";
```

Читання інформації з файлу відбувається аналогічно запису:

```
ifstream foo("ім'я_файлу") ;
char buffer[100];
foo >> buffer; //читає текст із файлу
```

Перед використанням класів потрібно включити заголовочний файл:

```
#include<fstream.h>
```

Коли розвантажується змінна потоку, потік, що відповідає файлу, закривається. Його можна закрити до того, як потік буде вичерпаний, використовуючи для цього функцію-член *close()*:

```
foo.close();
```

Зазвичай це роблять, якщо хочуть використовувати той самий потік для обслуговування декількох файлів. У цьому випадку потрібно закрити перший файл і відкрити інший файл за допомогою функції-члену *open()*:

```
foo.open("file. txt");
```

Для того, щоб дізнатися, чи залишилося ще що-небудь у файлі, потрібно скористатися функцією-членом *eof()*, що повертає значення "*істина*" при виявленні кінця файла.

Поради для читання і запису файлів:

• Найкраще не писати і не читати одночасно один і той самий файл. Хоча це цілком законно, але можна заплутатися в тому, яка частина файла читається,

а яка пишеться.

• При записуванні чисел до файлу за допомогою "<<" потрібно пам'ятати, що числа зберігаються в символьному форматі (як текст), але без прогалин між символами.

• Існує символ кінця файла, що повідомляє оператору ">>" про те, що файл закінчився.

• Рядки читаються доти, доки не буде зустрінутий роздільник (*eof* або \n*).* Якщо останній рядок у файлі не закінчується символом \n, то ознака кінця файла досягається відразу ж після завершення читання останнього рядку.

• При читанні чисел з потоку символ кінця файла за останнім числом відсутній. Тому може статися так, що програміст намагається прочитати на одне число більше, ніж їх є насправді у файлі або помістити символ кінця файлу в число. У цьому випадку ">>" нічого не буде робити, а, значить, число не зміниться. Отже, при читанні та записуванні до файлу, необхідна перевірка на кінець файлу.

Використання маніпуляторів

Існують деякі атрибути, вказуючи які можна змінити спосіб читання і запису у потоці. Ці атрибути називаються маніпуляторами потоку. Самі вони не дозволяють читати або записувати дані - вони лише впливають на засіб інтепретації даних при читанні та запису. Деякі маніпулятори використовуються з параметрами.

Маніпулятори введення/виведення визначаються в заголовочних файлах IOMANIP.H і IOSTREAM.H:

endl - Почати новий рядок; вивести буфер потоку.

ends - Вставити нульовий завершальний символ до рядку.

flush - Виконати виведення до потоку.

lock(ios &ir) - Заблокувати дескриптор файлу для посилки *ir* на потік введення/виведення.

resetiosflags(long f) - Очистити біти форматування.

setiosflags(long f) - Встановити біти форматування.

setfill(int c) - Використовувати символ «с» для заповнення при вирівнюванні.

setprecision(int n) - Встановити точність виведення значень з плаваючою комою рівною *n*

setw(int n) - Встановити ширину поля *n.*

unlock(ios &ir) — Розблокувати дескриптор файла для посилки *ir* потоків введення/виведення.

ws - Витягнути порожні символи.

Skipws - Ігнорувати порожній простір при введенні.

Left - "Притиснути" виведення до лівого боку поля.

Right - "Притиснути" виведення до правого боку поля.

Internal - " Притиснути" виведення після знаку або основи системи лічби.

Showbase - Використовувати індикатор основи системи лічби при виведенні.

Showpoint - Обов'язково показувати десяткову крапку при виведенні дійсного числа.

Uppercase - Привести до верхнього регістру виведення по основі системи лічби 16.

Showpos - Додати + при виведенні додатнього *int.*

Scientific - Використовувати форму 1.2345E2 для виведення дійсних значень.

Fixed - Використовувати форму 123.45 для виведення дійсних значень.

Untbuf - Флешувати всі потоки після виконання операції "<<".

Stdio - Флешувати потоки *stdout, stderr* після виконання операції "<<" .

Маніпулятори-функції потоків виведення повертають посилки на об'єкт *ostream*. Можна написати свої власні маніпулятори, включивши заголовочний файл IOSTREAM.H і визначивши функцію типу *&ostream*. Наприклад, визначимо маніпулятор дзвоника для потоку виведення:

```
ostream& bell(ostream& os)
 {
  return os << "\a"; // \a -  керуючий код дзвоника
```

```
}
```

Цю функцію можна використовувати в каскадному операторі виведення в потік:

```
cout << bell() << "Ding! ";
```

Можна викликати функцію-член потокового об’єкту для завдання вирівнювання. Наприклад, для виведення значення цілої змінної *v* із вирівнюванням по правому краю у восьми позиціях:

```
cout.width(8);
cout << v << '\n';
```

Модифікатор *width* впливає тільки на це виведене значення.

Символьні константи режиму відкриття потоку

In	Відкриття потоку для читання.
Out	Відкриття потоку для запису.
Ate	Перемотування до кінця файлу перед відкриттям.
App	Відкриття в режимі поповнення.
Trunc	Відкриття файлу з усіканням вже існуючого вмісту.
nocreate	Відкриття зазнає невдачі, якщо файл не існує.
norepleace	Відкриття зазнає невдачі, якщо файл існує.
binary	Відкриття у двійковому режимі.

Конструктори об'єктів класів *ifstream, ofstream, fstream* мають три форми, що перевантажуються, й розрізняються компілятором по числу і типу параметрів. Варіанти конструкторів:

• створення унікального об’єкту, що не з'єднаний із файлом: *ifstream(); fstream(); fstream();*

• створення унікального об’єкту, що з'єднується з відкритим файлом, заданим префіксом *fd*: *ifstream(int fd); ofstream(int fd); fstream(int fd);*

• створення унікального об’єкту, що з'єднується з відкритим файлом, заданим префіксом *fd*, а для обміну використовує буфер, на початок якого вказує *char** розміром *int* байт (третій параметр): *ifstream(int fd, char*, int)*

• створення унікального об’єкту, відкриття файлу, ім'я якого задає *char* fname*, а режим відкриття файлу - другий параметр, третій параметр задає захист файлу (file protection); за замовчуванням приймається значення, що

задається константою *filebuf::openprot*. Відкритий файл з'єднується з об'єктом:

```
ifstream (const char *fname, int, int = filebuf :: openprot);
```

Використовуючи потрібний конструктор, можна виконати посимвольний або блоко-орієнтований обмін із файлом. Обмін із файлом завжди буферизується і при створенні об'єктів класу *ifstream, ofstream, fstream* автоматично створюється об'єкт класу *filebuf*.

При організації обміну даними з файлом використовують функції-члени класу *filebuf*. Режимом відкриття файла керують символьні константи, описані для класу *ios*. В усіх випадках, коли прикладна програма хоче змінити встановлений режим доступу до файлу за замовчуванням, треба виконати створення об'єктів класу *ifstream, ofstream, fstream* із заданим другим аргументом. При цьому константи режиму можуть об'єднуватися з операцією порозрядного логічного *АБО*.

Наприклад, якщо файл *c:\tools\text.tmp* потрібно відкрити для запису в режимі поповнення без створення нового файлу, то в програмі створюється об'єкт класу *ofstream* таким чином:

```
ofstream                                my_output("c:\tools\text.
tmp",ios::noreplace,ios::app);
```

Якщо файл успішно відкритий, то можна виконати читання і запис, керування покажчиком потоку функціями-членами базових класів *istream* і *ostream*. Для визначення стану файлового введення/виведення використовують функції базового класу *ios*.

Рядково-орієнтоване введення-виведення

Введення рядків виконує оператор ">>". Наприклад:

```
char string[M];
cin >> string;
```

Проте, його використання може викликати небажані сторонні ефекти, якщо число введених символів перевищує обсяг зарезервованих для рядка символів. Для введення рядка з контролем числа прийнятих символів можна використовувати функцію-член класу *istream get(char*,int,char)*. Наприклад,

для прийому рядка символів, для якого зарезервовано *M* байтів, треба зробити так:

```
char string[M];
cin.get(string,M-1,'\n');
```

Якщо введено більш ніж *M-1* символів, вони залишаються в буфері введення і будуть передані прикладній програмі при наступному використанні оператора ">>". Зокрема, завжди залишиться в буфері введення символ переведення рядка. Щоб усунути сторонні ефекти, при наступних операціях введення треба звільнити буфер введення. Можливе рішення - перемістити покажчик потоку на кінець, використовуючи функцію-член *seekg()*:

```
cin.seekg(0,ios::end);
```

Виведення рядка виконує оператор "<<". Якщо необхідно вивести тільки частину рядка, використовується функція-член класу *ostream write()*. Наприклад:

```
char string[M];
cout.write(string,10);
```

виведе 10 перших символів рядку *string*.

Приклад. Програма приймає з клавіатури довільне число рядків і записує їх до файлу. Ім'я файлу задається при запуску програми першим параметром командного рядка. Робота програми завершується при одержанні з клавіатури умови *EOF*,

```
#include<fstream. h>
int main(int argc, char **argv)
 {
  char ch;
  if(argc<2)
   {
    cerr << "\a Використання програми:\n";
    cerr << argv[0] << "ім'я файлу признач.  \n";
    return 255;
    }
//Спроба створити об'єкт ofstream
ofstream output_to_file(argv[1]);
if(! output_to_file)
```

```
  {
   cerr << "\a Помилка відкриття" << argv[1];
   cerr << "для запису\n";
   return 254;
   }
while(!cin.eof())
  {
    cin.get(ch);
    output_to_file.put(ch);
  }
 output_to_file.close();
 return 0;
}
```

Перевантаження оператора виведення в потік

Зазвичай потоки виведення можуть підтримувати лише базові типи даних. За допомогою перевантаження оператора виведення в потік << можна змусити його виводити об'єкти класів.

Приклад: у програмі перевантажується оператор виведення в потік.

```
class TPoint
{
 private:
  int x,y;
 public:
  TPoint()
   {
    x=y=0;
   }
  TPoint(int xx, int yy)
   {
    x=xx;
    y=yy;
   }
 void PutX(int xx)
  {
```

```
    x=xx;
   }
  void PutY(int yy)
   {
    y=yy;
   }
  int GetX(void)
  {
   return x;
  }
  int GetY(void)
  {
   return y;
  }
  friend ostream& operator<<(ostream &os, TPoint &p);
};
main()
{
 TPoint p;
 cout << p << '\n';
 p.Put(100);
 p.Put(200);
 cout << p << '\n';
 return 0;
}
ostream& operator<<(ostream &os, TPoint &p)
{
 os << "x==" << p.x << ",y==" << p.y;
 return os;
}
```

Тут за допомогою дружньої функції-члена *operator<<()* перевантажується оператор виведення в потік. Ця функція повертає значення типу *ostream&* і в ній оголошені два параметри: *os* типу *ostream&* і посилка *p* на об'єкт класу *TPoint*. Оскільки функція *operator<<()* є другом класу *TPoint*, у ній можна зробити безпосередній доступ до даних-членів *x* і *y* того об’єкту, на який посилається *p*. Функція повертає посилку *os*. Запуск програми призведе до

таких результатів:

x==0, y==0

x==100, y==200

Тобто функція *operator<<()* повертає посилку на *ostream.* Можна виводити декілька об'єктів в одному операторі виведення в потік:

cout << p1 << “,” << p2 << “,” << p3;

Також можливе і перевантаження оператора введення з потоку.