



## Bài 6: Luồng vào ra

---



## Nội dung

---

1. Khái niệm luồng. Các lớp luồng.
2. Luồng xuất và các phương thức trên luồng xuất.
3. Luồng nhập và các phương thức trên luồng nhập.
4. Giới thiệu thư viện iomanip.
5. Review
6. Bài tập



## Ôn lại

1. Phương thức ảo
2. Đa hình
3. Huỷ tử ảo
4. Lớp cơ sở trừu tượng

3

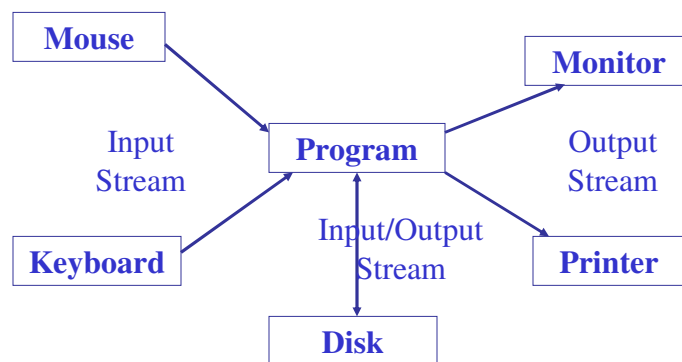


## Luồng/Dòng (STREAMS)

- Luồng
  - Luồng là một môi trường trung gian để chương trình giao tiếp với các thiết bị.
  - Một dãy các ký tự (dãy byte dữ liệu), kết thúc bởi ký hiệu *end\_of\_file*
- Nhiệm vụ của luồng
  - các luồng có nhiệm vụ nhận thông tin từ thiết bị hoặc gửi thông tin ra các thiết bị. Do đó khi cần làm việc với các thiết bị ta chỉ cần làm việc với các luồng.
- Luồng nhập:
  - Một dãy byte đổ vào các biến được gọi là luồng nhập (từ bàn phím, đĩa... vào bộ nhớ)
- Luồng xuất:
  - Một dãy byte đổ ra khỏi chương trình được gọi là luồng xuất (từ bộ nhớ ra màn hình, máy in...).

4

## Minh họa luồng vào ra



5

## Thư viện cho luồng vào/ra

- **Thư viện `iostream`**
  - Có nhiều header file với nhiều chức năng vào ra
- **`<iostream.h>`**
  - vào chuẩn – Standard input (`cin`)
  - ra chuẩn – Standard output (`cout`)
  - dòng báo lỗi không có bộ nhớ đệm – Unbuffered error (`cerr`)
  - dòng báo lỗi có dùng bộ nhớ đệm – Buffered error (`clog`)
- **`<iomanip.h>`**
  - các stream manipulator (có tham số) để định dạng I/O
- **`<fstream.h>`**
  - các thao tác xử lý file

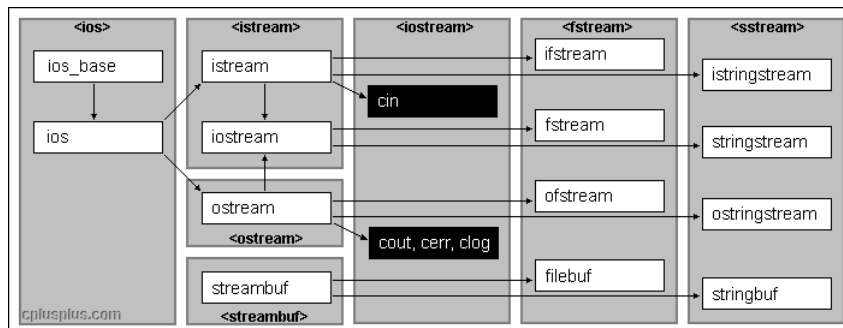
6

## Các lớp nhập/xuất

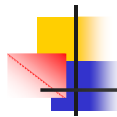
- Lớp ios: là lớp cơ sở của các lớp nhập/xuất.
- Lớp istream và lớp ostream được dẫn xuất từ lớp ios và được chuyên biệt hóa để quản lý các thao tác nhập/xuất
- Lớp iostream được dẫn xuất từ hai lớp istream và ostream bao gồm các phương thức nhập/xuất cho bàn phím và màn hình
- Lớp fstream chứa các phương thức nhập/xuất tệp tin.
- Lớp streambuf: quản lý bộ đệm, bao gồm các phương thức có khả năng làm đầy, làm rỗng, vét bộ đệm và các thao tác khác liên quan đến bộ đệm

7

## Các lớp nhập/xuất



8



## Các đối tượng nhập xuất

- << và >>
  - các toán tử chèn và tách dòng
- cin
  - đối tượng **istream**
  - điều khiển thao tác nhập từ thiết bị chuẩn(như bàn phím).
  - Ví dụ: `cin >> grade;`
    - trình biên dịch tự xác định kiểu của `grade`
    - gọi toán tử thích hợp (đã được định nghĩa sẵn)
    - không cần thông tin thêm về kiểu dữ liệu
- cout
  - điều khiển các thao tác xuất đến thiết bị chuẩn(như màn hình).
  - đối tượng **ostream**
  - Ví dụ: `cout << grade;`
    - cũng như với `cin`, không cần thêm thông tin về kiểu

9



## Các đối tượng nhập xuất

- cerr
  - đối tượng **ostream**
  - điều khiển các thao tác xuất(không sử dụng bộ đệm) đến thiết bị báo lỗi chuẩn(màn hình).
- clog
  - đối tượng **ostream**
  - điều khiển thao tác thông báo lỗi(có sử dụng bộ đệm) đến thiết bị báo lỗi chuẩn(màn hình).

10



## Đối tượng cerr

```
1. #include <iostream.h>
2. int main(){
3.     int a, b;
4.     cout<<"Nhập a";
5.     cin>>a;
6.     cout<<"Nhập b";
7.     cin>>b;
8.     if (b == 0) cerr<<"Lỗi chia cho 0"<<endl;
9.     else cout<<"a/b="<<a/b<<endl;
10.    return 0;
11. }
```

11



## Luồng xuất (Output Stream)

- Xuất dữ liệu với cout.
- Thao tác xuất dữ liệu tăng.
- Xuất các biến kiểu char\*.
- Xuất dữ liệu kiểu ký tự với hàm thành viên put, và tăng put.

12



## Luồng xuất (Output Stream)

- Xuất dữ liệu với cout

- cout<< bieu\_thuc;

- trong đó bieu\_thuc

- một biến
    - một hằng
    - một biểu thức chứa biến hằng, hàm

- Ví dụ:

- cout<<a;
  - cout<<PI; cout<<3.14;
  - cout<<(a + b\*sin(c));

13



## Luồng xuất (Output Stream)

- Xuất dữ liệu theo tầng

- Kiểu trả về của toán tử xuất của lớp ostream
  - ostream& operator <<(tham\_số);
  - Cụ thể: ostream& operator <<(int&);  
ostream& operator <<(float&);  
ostream& operator <<(char&);  
ostream& operator <<(long&);

- Do đó có thể viết:

- cout<<a<<b<<endl;
  - Hoặc (((cout<<a)<<b)<<endl);

14

## Luồng xuất (Output Stream)

- Xuất các biến kiểu `char*`
  - C++ tự động xác định kiểu dữ liệu
    - in giá trị của một `char *`: địa chỉ bộ nhớ của ký tự đầu tiên
  - Rắc rối
    - **toán tử** `<<` được định nghĩa chống để in xâu kết thúc bằng null
    - cách giải quyết: đổi thành `void *`
      - sử dụng khi in giá trị của một con trỏ
      - in dưới dạng một số cơ số 16
  - Ví dụ:

```
char *str = "Test!"; hoặc char str[10]="Test!";
cout<<str; => Test!
cout<<(void*)str;=> Địa chỉ của str[0]
cout<<static_cast<void*>(str); => Địa chỉ của str[0]
```

15

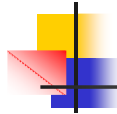
## Ví dụ xuất các biến kiểu `char*`

```
1 // ví dụ xuất dữ liệu kiểu char *
2 // In địa chỉ của biến kiểu char *
3 #include <iostream>
4 using namespace std;
5
6 int main()
7 {
8     char *word = "test";
9
10    cout << "Value of word is: " << word << endl
11        << "Value of static_cast< void * >( word ) is: "
12        << static_cast< void * >( word ) << endl;
13
14    return 0;
15
16 }
```

Để in giá trị của con trỏ, ta phải đổi sang kiểu `void *`. Nếu không, chương trình sẽ in xâu ký tự.

16





## Luồng xuất (Output Stream)

- Xuất dữ liệu kiểu ký tự với hàm put và write
  - Toán tử xuất hỗ trợ hai hàm là put và write
- Hàm put: dùng để ghi một ký tự lên thiết bị xuất. Cũng như toán tử xuất, hàm put có thể được gọi liên tiếp.
  - Ví dụ  
`cout.put('H');` Hoặc  
`cout.put('H').put('e').put('l').put('l').put('o').put('\n');`
  - Có thể sử dụng giá trị bằng số (mã ASCII)
    - `cout.put( 65 );`
    - in ký tự 'A'
- Hàm write
  - Cú pháp: `ostream& write(const char*, int);`
  - Ví dụ: `cout.write("Hello",5);`

17



## Luồng nhập (Input Stream)

- Thao tác nhập dữ liệu
  - Nhập các biến kiểu dữ liệu chuẩn
  - Nhập chuỗi ký tự
  - Nhập dữ liệu theo tầng
- Một số phương thức của cin
  - Nhập một ký tự(`get`)
    - `Get` không tham số
    - `Get` có tham số
  - Nhập chuỗi từ bàn phím(`getline`)
  - Các hàm `peek` và `putback`

18



## Luồng nhập (Input Stream)

- Nhập dữ liệu cho các kiểu biến chuẩn
  - Cú pháp: `cin>>Tên_biến;`  
Trong đó: Tên\_biến: kiểu char, int, long, float, double.
- Toán tử `>>`
  - Thường bỏ qua các ký tự trắng (blank, tab, newline)
  - Trả về 0 khi gặp EOF

19



## Luồng nhập (Input Stream)

- Nhập chuỗi ký tự
  - Chuỗi ký tự: là một mảng các phần tử kiểu char được kết thúc bởi ký tự null(`'\0'`).
  - Nhập dữ liệu cho chuỗi ký tự:  
`cin>>biến_chuỗi_ký_tự;`
  - Ví dụ:  
`char str[256];`  
`cin>>str;`
  - Chú ý: Không thể nhập dấu cách trong chuỗi này!

20



## Luồng nhập (Input Stream)

- Nhập với hàm `get()` không có tham số
  - `cin.get()`
  - trả về một ký tự từ dòng (kể cả ký tự trắng)
    - trả về **EOF** nếu gặp end-of-file (đánh dấu kết thúc dữ liệu vào )
- Ví dụ

```
int main(){
    char ch;
    while((ch=cin.get()) != EOF)
    {
        cout<<"ch: " << ch;
    }
}
```
- hàm `cin.eof()`
  - trả về 1 (**true**) nếu đã gặp EOF

21



## Luồng nhập (Input Stream)

- Nhập với hàm `get()` có 1 tham số
  - nguyên mẫu hàm: `istream& get(char&);`
- Ví dụ:

```
int main(){
    char a, b, c;
    cout<<"Nhập 3 ký tự a, b, c"
    cin.get(a).get(b).get(c);
    cout<<"a: "<<a<<" b:"<<b<<" c:"<<c<<endl;
    return 0;
}
```


22



## Luồng nhập (Input Stream)

- Nhập với hàm `get()` có 3 tham số
  - Nguyên mẫu hàm: `istream& get(unsigned char*, int len, char='\n');`
- Giải thích
  - `unsigned char*`: chuỗi lưu kết quả
  - `int len`: số ký tự tối đa có thể nhập
  - `char = '\n'`: ký tự kết thúc, ngầm định là Enter
- Hàm `get` không lấy ký tự xuống dòng(Enter) trong bộ đệm

23



## Luồng nhập (Input Stream)

- Hàm `GetLine()`
  - Nguyên mẫu hàm:  
`istream& getline(unsigned char*, int len, char='\n');`
- Giải thích:
  - `unsigned char*`: chuỗi lưu kết quả
  - `int len`: số ký tự tối đa có thể nhập
  - `char = '\n'`: ký tự kết thúc, ngầm định là Enter
- Hàm `getline` lấy cả ký tự xuống dòng(Enter) trong bộ đệm

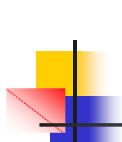
24



## Một số hàm xử lý chuỗi ký tự

- Thư viện: <string.h>
- Một số hàm xử lý về chuỗi:
  - `char* strcpy(char*dest, const char*source);`
  - `int strcmp(const char*, const char*);`
  - `char* strstr(const char*, const char*);`
  - `int strlen(const char*);`

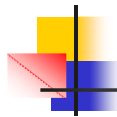
25



## Các hàm thành viên peek, putback và ignore của istream

- **ignore ()**
  - lấy các ký tự khỏi luồng (loại bỏ ký tự ra khỏi bộ nhớ đệm), mặc định là 1 ký tự.
  - dừng khi gặp ký tự phân cách
    - phân cách mặc định là `EOF`
- **putback ()**
  - đẩy ký tự vừa đọc được bằng `get ()` trở lại luồng
- **peek ()**
  - trả về ký tự tiếp theo trong dòng nhưng không lấy ra khỏi luồng

26



## Định dạng nhập xuất dữ liệu

- Thư viện: <iomanip.h>
- Chứa các thông tin định dạng nhập xuất
  - Độ chính xác dấu phẩy động(precision, setprecision).
  - Trường độ rộng(setw, width).
  - căn trái/phải/giữa (left/right/internal)
  - ký tự chèn vào các vị trí còn trống (setfill)
  - ....

27



## Đặt độ rộng xuất dữ liệu

- Đặt độ rộng xuất dữ liệu
  - `cout.width(độ_rộng);`
  - `setw(độ_rộng);`
- Ví dụ:
- ```
int main(){  
    int a = 12; b = 345; // độ rộng thực của a là 2, của b là 3  
    cout << a; // chiếm 2 cột màn hình  
    cout.width(7); // đặt độ rộng giá trị in tiếp theo là 7  
    cout.fill('*') //thay thế ký tự trống bằng *  
    cout << b; // b in trong 7 cột với 4 dấu cách đứng trước  
}
```

==>Kết quả: 12\*\*\*\*345

28



## Cờ và các chỉ thị định dạng

- Các cờ định dạng.
  - `cout.setf(danh sách cờ);` // Bật các cờ trong danh sách
  - `cout.unsetf(danh sách cờ);` // Tắt các cờ trong danh sách
- Dấu chấm thập phân.
  - `ios::showpoint:` in đủ n chữ số lẻ của phần thập phân, nếu tắt (ngầm định) thì không in các số 0 cuối của phần thập phân
- Căn lề
  - `ios::left`
  - `ios::right`
- Đặt ký tự lấp đầy
  - `fill`
  - `setfill`

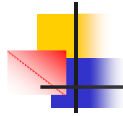
29



## Ví dụ về cờ và định dạng

```
#include <iostream.h>
#include <iomanip.h>
int main()
{
    int a = 12.3; b = -345.678; // độ rộng thực của a là 4, của b là 8
    cout << a; // chiếm 4 cột màn hình
    cout.width(10); // đặt độ rộng giá trị in tiếp theo là 10
    cout.fill('*'); // dấu * làm ký tự đệm
    cout.precision(2); // đặt độ chính xác đến 2 số lẻ
    cout.setf(ios::left); // bật cờ ios::left
    cout << b; // kết quả: 12.3-345.68***
    cout.setf(ios::right); // bật cờ ios::right
    cout << b; // kết quả: 12.3***-345.68
    return 0;
}
```

30



## Review

---

1. Khái niệm luồng. Các lớp luồng.
2. Luồng xuất và các phương thức trên luồng xuất.
3. Luồng nhập và các phương thức trên luồng nhập.