

Ôn tập C++, lớp đối tượng và vẽ sơ đồ UML

CSC10003 – Phương pháp lập trình hướng đối tượng

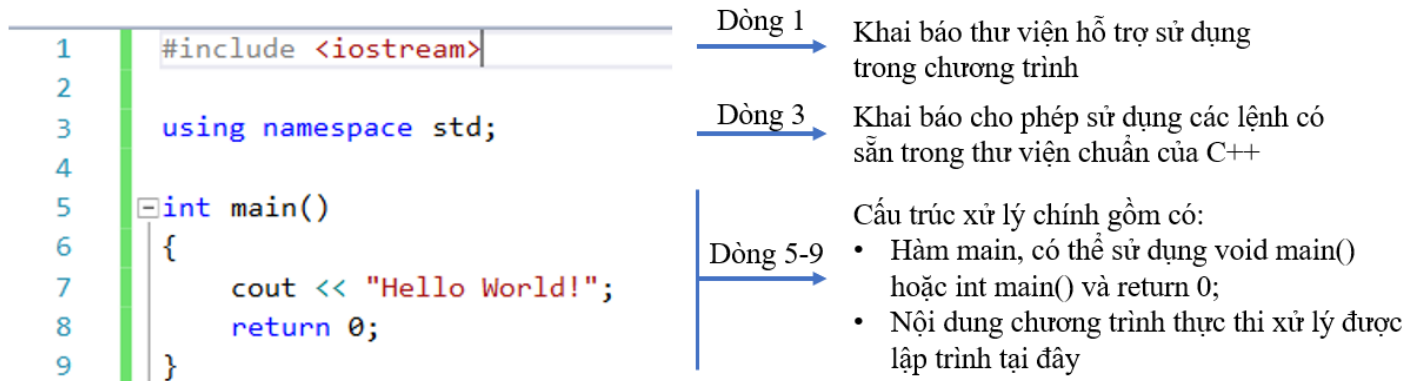
Nội dung thực hành lần này sẽ tập trung ôn tập kiến thức về lập trình C++ cùng với việc tìm hiểu, cài đặt lớp đối tượng và hướng dẫn vẽ sơ đồ UML

1. Ôn tập về C++

1.1. Cấu trúc của chương trình C++

Cấu trúc của một chương trình C++ cơ bản gồm các phần theo thứ tự sau:

- Phần khai báo thư viện nằm ở đầu chương trình với các câu lệnh `#include`
- Phần khai báo cho phép sử dụng các lệnh có sẵn trong thư viện chuẩn của C++ để tiện sử dụng khi lập trình
- Các hàm con xử lý
- Hàm `main` nằm ở dưới cùng để xử lý thực thi chương trình



Khai báo thư viện với cú pháp

```
#include <tên thư viện>
```

- Một số thư viện thường hay sử dụng:

- `#include<iostream>` : sử dụng cho nhập xuất ra màn hình console
- `#include<math>` : sử dụng cho việc xử lý tính toán số học
- `#include<string>` : sử dụng cho việc xử lý chuỗi
- `#include<vector>`: sử dụng cho việc lưu trữ dữ liệu dạng vector
- `#include<ctime>` : sử dụng cho việc xử lý thời gian

Một số lưu ý:

- Nên theo **đúng thứ tự** cấu trúc xử lý chương trình, đặc biệt là các hàm con phải nằm trước hàm `main`, do chương trình thực thi tuần tự từ trên xuống dưới. Nếu hàm `main` được gọi trước các hàm con thì các hàm con sử dụng trong hàm `main` sẽ không được tìm thấy và chương trình báo lỗi.

- Các lệnh xử lý trong chương trình đa phần nằm ở tập tin .cpp, ngoài ra còn có tập tin .h dùng để khai báo trước các hàm và lớp sẽ sử dụng. Do đó có thể sử dụng tập tin .h để khai báo trước cấu trúc của chương trình.
- Nhấn Ctrl + F5 để thực thi chương trình và xem kết quả, nhấn F10 để debug chương trình theo từng dòng bắt đầu từ hàm main, nhấn F11 để vào bên trong dòng lệnh để debug.

1.2. Nhập và xuất ra màn hình Console trong C++

- ✚ C++ cung cấp các lệnh có sẵn để hỗ trợ việc nhập xuất ra màn hình, dựa trên thư viện chuẩn của C++, để sử dụng, đầu chương trình cần phải khai báo ở đầu chương trình

```
#include<iostream>

using namespace std;
```

- ✚ Các lệnh nhập xuất sử dụng trong C++ :

- cin : nhập dữ liệu, sử dụng toán tử >>
- cout : xuất dữ liệu, sử dụng toán tử <<

- ✚ Ghi nhớ tránh nhầm lẫn : **Cin đi vào biến, biến đi ra Cout (Cin >>, Cout <<)**

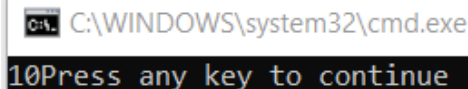
- ✚ Cú pháp sử dụng để nhập dữ liệu vào biến :

```
int variable;
cin >> variable;
```

- Màn hình sẽ chờ người dùng nhập dữ liệu vào
- Dữ liệu sau khi nhập, nhấn Enter để hoàn tất quá trình nhập

- ✚ Cú pháp sử dụng để xuất dữ liệu ra màn hình console :

```
int variable = 10;
cout << variable;
```

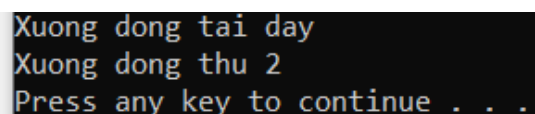


```
C:\WINDOWS\system32\cmd.exe
10Press any key to continue . . .
```

- ✚ Các ký tự đặc biệt :

- Ký tự xuống dòng : “\n” hoặc endl
- Ký tự cách dòng tab : “\t”
- Ký tự kết thúc chuỗi : “\0”

```
cout << "Xuong dong tai day" << endl;
cout << "Xuong dong thu 2\n";
```



```
Xuong dong tai day
Xuong dong thu 2
Press any key to continue . . .
```

2. Lập trình hàm và lập trình hướng đối tượng

2.1. Lập trình hàm – lập trình hướng thủ tục

- ✚ Một chương trình khi xử lý được chia thành nhiều hàm, trong đó mỗi hàm xử lý có thể có hàm con
- ✚ Mỗi hàm đều có tên hàm, tham số đầu vào hàm và kết quả trả về
- ✚ Đầu vào có thể truyền theo dạng tham trị hoặc tham chiếu
 - Truyền tham trị : giá trị biến sau khi ra khỏi hàm sẽ không thay đổi
 - Truyền tham chiếu : giá trị biến sau khi ra khỏi hàm có thể bị thay đổi

```
//Hàm cộng hai số nguyên
//Biến a truyền tham trị, biến b truyền tham chiếu
int Cong2SoNguyen(int a, int &b)
{
    return a + b;
}
```

- ✚ Đặc điểm của lập trình theo hướng thủ tục :
 - Dựa theo chức năng làm trọng tâm : mỗi hàm xử lý thực hiện một số hành động hay thao tác nào đó.
 - Chương trình gọi các hàm xử lý tuần tự, truyền dữ liệu vào và trả về kết quả

2.2. Lập trình hướng đối tượng

- ✚ Thay vì tập trung vào xử lý dựa theo chức năng làm trọng tâm, lập trình hướng đối tượng tập trung vào đối tượng cần xử lý, trong đối tượng đó sẽ có khả năng thực hiện chức năng gì và có những thao tác, thuộc tính nào có liên quan
- ✚ Mỗi lớp đối tượng được dùng để mô tả một nhóm các đối tượng cùng loại có cùng một cách ứng xử trong thế giới thực.
- ✚ Các thành phần trong lớp đối tượng (**class**) :
 - Tên lớp : mỗi lớp sẽ có tên duy nhất phân biệt với lớp khác trong cùng phạm vi
 - Thành phần dữ liệu hay thuộc tính (**attributes**): các trường để lưu dữ liệu của lớp đối tượng
 - Các phương thức của lớp (**methods**): là các hàm xử lý viết riêng cho đối tượng của lớp.
- ✚ Cài đặt lớp đối tượng, cần phải xác định các thành phần trong lớp đối tượng
- ✚ Mỗi thuộc tính và phương thức sẽ có **tầm vực truy cập (scope)**

- **Public** : cho phép truy xuất thuộc tính và phương thức từ bên ngoài lớp
- **Private** : chỉ cho phép truy xuất, thay đổi thuộc tính, phương thức ở bên trong nội bộ của lớp đó
 - Nếu không khai báo tầm vực, **mặc định sẽ là private**

```
#include<iostream>

using namespace std;

//Tên class: PhanSo
class PhanSo
{

    //Phần khai báo thuộc tính
    //Nếu để mặc định thì sẽ là private
    int tuso;
    int mauso;

    //Phần khai báo phương thức bên trong lớp đối tượng
public:
    voidNhapPhanSo()
    {
        cout << "Moi nhap tu so: ";
        cin >> tuso;
        cout << "Moi nhap mau so: ";
        cin >> mauso;
    }

    //Khai báo xử lý phương thức bên ngoài đối tượng
    void XuatPhanSo();
};

//Xử lý phương thức Xuất phân số bên ngoài đối tượng
void PhanSo::XuatPhanSo()
{
    cout << "Tu so: " << tuso << endl;
    cout << "Mau so: " << mauso << endl;
}

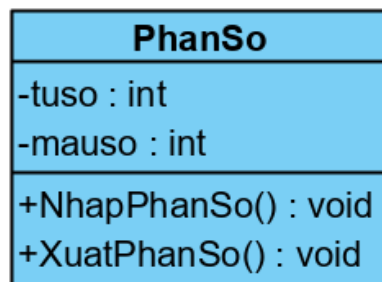
void main()
{
    //Sử dụng lớp trong hàm main
    PhanSo a;
    //Đối tượng a là Phân Số gọi phương thức Nhập
    a.NhapPhanSo();
}
```

Ví dụ về khai báo lớp đối tượng Phân Số gồm có tử số và mẫu số

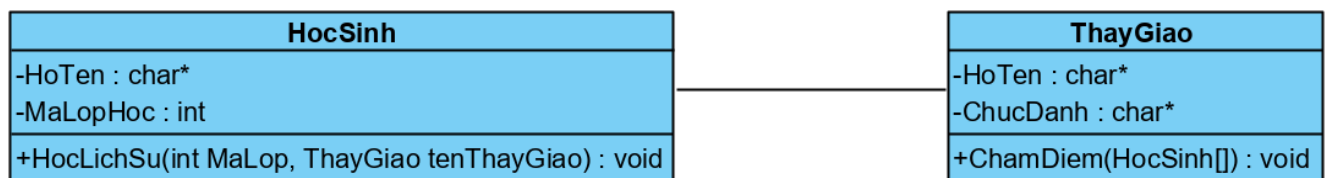
- ✚ Có 2 cách để khai báo và xử lý phương thức của lớp đối tượng :
 - Khai báo và xử lý trực tiếp bên trong đối tượng đó
 - Khai báo trước bên trong đối tượng, bên ngoài xử lý riêng
- ✚ Sử dụng đối tượng : khai báo trước thông tin đối tượng, gọi các phương thức xử lý tương ứng (xem ví dụ trong mã nguồn phía trên)

3. Vẽ sơ đồ lớp UML (Class diagram)

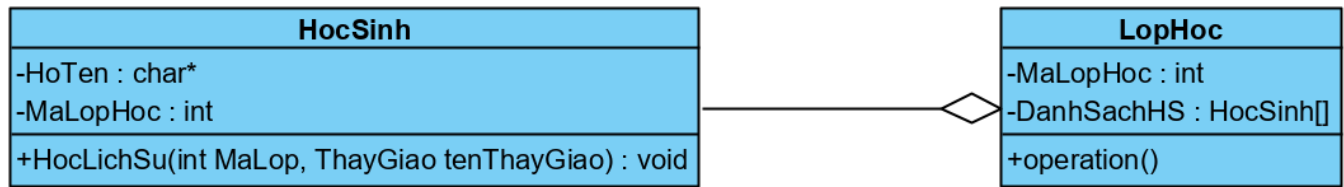
- ✚ Ngôn ngữ UML (Unified Modeling Language) là một hệ thống ký hiệu dùng cho việc mô tả hay là mô hình hóa các đối tượng trong chương trình được viết bằng các ngôn ngữ lập trình hướng đối tượng, Ngôn ngữ này mô tả nhiều khía cạnh khác nhau của đối tượng và đưa ra một cái nhìn tổng quan về toàn bộ chương trình.
- ✚ Công cụ hỗ trợ vẽ UML miễn phí: **Visual Paradigm Community**
- ✚ Ký hiệu của lớp đối tượng: gồm 1 hình chữ nhật, trong đó có 3 khung chứa thông tin tương ứng theo thứ tự:
 - Tên lớp
 - Thuộc tính: kiểu dữ liệu
 - Phương thức: tham số đầu vào và kiểu dữ liệu trả về



- ✚ Tầm vực của thuộc tính và phương thức:
 - Nếu trước thuộc tính và phương thức có dấu trừ (-) thì sẽ là private
 - Nếu trước thuộc tính và phương thức có dấu cộng (+) thì sẽ là public
- ✚ Mối liên hệ: mỗi lớp đối tượng sẽ có thể có mối quan hệ với đối tượng khác
 - Association: đối tượng có mối liên hệ với nhau nhưng không có tính sở hữu



- Aggregation: đối tượng có mối liên hệ với nhau và có tính sở hữu, trong một đối tượng có thuộc tính là một hoặc nhiều đối tượng khác (quan hệ chứa trong, trong đó lớp chứa đối tượng khác sẽ có hình thoi khi vẽ liên kết)



✚ Trước khi vẽ sơ đồ UML, cần phải xác định:

- Có bao nhiêu lớp?
- Mỗi lớp có phương thức và thuộc tính gì?
- Thuộc tính có kiểu dữ liệu gì? Có tầm vực thế nào?
- Phương thức bao nhiêu tham số đầu vào? Có kiểu trả về thế nào? Có tầm vực thế nào?
- Các lớp có mối quan hệ với nhau hay không?
 - Nếu lớp này là một thuộc tính của lớp khác (chứa trong), có thể sẽ là quan hệ Aggregation
 - Nếu lớp này chỉ là tham số đầu vào hay kết quả trả về mà không phải là một thuộc tính của lớp khác: có thể là Association

4. Bài tập thực hành

Bài tập 1: Viết chương trình C++ cài đặt đối tượng phân số, trong đó phân số được định nghĩa gồm có tử số và mẫu số với kiểu dữ liệu là số nguyên. Sinh viên hãy cài đặt các thao tác trên đối tượng phân số này:

- Cộng, trừ, nhân, chia 2 phân số
- Nhập phân số từ console
- Xuất phân số ra console theo 2 dạng:
 - Xuất theo định dạng tử số / mẫu số
 - Xuất theo định dạng kết quả phép chia của tử số và mẫu số (số thập phân)

Bài tập 2: Viết chương trình C++ cài đặt đối tượng điểm trong không gian Oxy, trong đó một điểm được định nghĩa gồm có tọa độ x và tọa độ y đều là số thực. Sinh viên hãy cài đặt các thao tác trên đối tượng điểm này:

- Nhập và xuất điểm ra màn hình console
- Tính khoảng cách giữa 2 điểm trong không gian Oxy, xuất ra thông tin khoảng cách

Bài tập 3: Viết chương trình C++ cài đặt đối tượng hình chữ nhật trong không gian Oxy gồm có 4 đỉnh (theo thứ tự kim đồng hồ, đỉnh đầu tiên ở góc trái trên), trong đó mỗi đỉnh của hình chữ nhật là 1 điểm (x, y) trong không gian Oxy. Sinh viên hãy cài đặt các thao tác trên đối tượng hình chữ nhật này:

- Nhập và xuất thông tin hình chữ nhật ra màn hình console
- Kiểm tra xem hình chữ nhật có hợp lệ hay không dựa trên thông tin các đỉnh, xuất ra màn hình kết quả kiểm tra hợp lệ hay không hợp lệ.
- Tính chu vi và diện tích hình chữ nhật nếu hình chữ nhật hợp lệ.
- Vẽ sơ đồ lớp UML cho bài tập này