



Nguyễn Việt Nam Sơn
nvnamson@gmail.com

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Constructor & Destructor Getter & Setter

1

NỘI DUNG BÀI HỌC

2

1. PHƯƠNG THỨC KHỞI TẠO (CONSTRUCTOR)

2. PHƯƠNG THỨC PHÁ HỦY (DESTRUCTOR)

3. GIẢI ĐÁP NHỮNG LƯU Ý

4. GETTER & SETTER

5. BÀI TẬP ỨNG DỤNG TẠI LỚP

6. BÀI TẬP VỀ NHÀ

1. PHƯƠNG THỨC KHỞI TẠO (CONSTRUCTOR)

3

SỰ CẦN THIẾT CỦA CONSTRUCTOR ?

Chuyện gì sẽ xảy ra nếu như người lập trình sử dụng 1 đối tượng trong khi đối tượng đó chưa hề được khởi tạo ra ?

Trả lời: Các thành phần thuộc tính của đối tượng sẽ nhận giá trị rác từ vùng nhớ (do còn sót lại trước đó) hoặc phải nhận 1 giá trị không mong muốn nào đó.

=> Mục đích của phương thức khởi tạo đó là nhằm khởi tạo các thành phần dữ liệu (thuộc tính) của đối tượng.

1. PHƯƠNG THỨC KHỞI TẠO (CONSTRUCTOR)

4

ĐẶC ĐIỂM CỦA CÁC PHƯƠNG THỨC KHỞI TẠO

1. Tên phương thức trùng với tên lớp và không trả về giá trị (nhưng có thể có tham số nhé).
2. Tự động được gọi thực thi ngay khi đối tượng của lớp đó được tạo lập ra (có bao nhiêu đối tượng được tạo ra thì sẽ có bấy nhiêu lần phương thức đó được gọi thực thi).
3. Có thể có nhiều phương thức khởi tạo bên trong 1 lớp (cơ chế nạp chồng overloading do phân biệt nhau qua tham số truyền vào).

1. PHƯƠNG THỨC KHỞI TẠO (CONSTRUCTOR)

5

CÁC LOẠI PHƯƠNG THỨC KHỞI TẠO

1. Phương thức khởi tạo mặc định (Default Constructor)
2. Phương thức khởi tạo có tham số truyền vào (Constructor With Paramater)
3. Phương thức khởi tạo sao chép (Copy Constructor)

1. PHƯƠNG THỨC KHỞI TẠO (CONSTRUCTOR)

6

PHƯƠNG THỨC KHỞI TẠO MẶC ĐỊNH (DEFAULT CONSTRUCTOR)

- Là phương thức khởi tạo không có tham số đầu vào.
- Mặc định trình biên dịch sẽ phát sinh sẵn phương thức khởi tạo mặc định nếu như bản thân lớp đó không có bất kỳ phương thức khởi tạo nào khác. (**Lưu ý số 1**)

1. PHƯƠNG THỨC KHỞI TẠO (CONSTRUCTOR)


7

PHƯƠNG THỨC KHỞI TẠO MẶC ĐỊNH (DEFAULT CONSTRUCTOR)

File: HocSinh.h

```
#pragma once
#include <iostream>
#include <string>
using namespace std;

class HocSinh
{
private:
    string HoTen;
    float DiemThi;
public:
    HocSinh(void);
    ~HocSinh(void);
};
```



Constructor - Destructor - Getter - Setter

File: HocSinh.cpp

```
#include "HocSinh.h"
↓
HocSinh::HocSinh(void)
{
    HoTen = "Son dep trai";
    DiemThi = 9.9;
}

HocSinh::~~HocSinh(void)
{
}
```

1. PHƯƠNG THỨC KHỞI TẠO (CONSTRUCTOR)

8

PHƯƠNG THỨC KHỞI TẠO NHẬN THAM SỐ ĐẦU VÀO (CONSTRUCTOR WITH PARAMATER)

Là phương thức khởi tạo nhận tham số đầu vào “tùy ý” do người dùng định nghĩa, sẽ lấy đó làm dữ liệu tạo cho đối tượng.

1. PHƯƠNG THỨC KHỞI TẠO (CONSTRUCTOR)


9

PHƯƠNG THỨC KHỞI TẠO NHẬN THAM SỐ ĐẦU VÀO (CONSTRUCTOR WITH PARAMATER)

File: HocSinh.h

```
#pragma once
#include <iostream>
#include <string>
using namespace std;

class HocSinh
{
private:
    string HoTen;
    float DiemThi;
public:
    HocSinh(void);
    HocSinh(string, float);
    ~HocSinh(void);
};
```



Constructor - Destructor - Getter - Setter


File: HocSinh.cpp

```
#include "HocSinh.h"

HocSinh::HocSinh(void)
{
    HoTen = "Son dep trai";
    DiemThi = 9.9;
}

HocSinh::HocSinh(string hoten, float diem)
{
    HoTen = hoten;
    DiemThi = diem;
}

HocSinh::~~HocSinh(void)
{
}
```



1. PHƯƠNG THỨC KHỞI TẠO (CONSTRUCTOR)

PHƯƠNG THỨC KHỞI TẠO SAO CHÉP (COPY CONSTRUCTOR)

Là phương thức khởi tạo nhận tham số đầu vào là 1 đối tượng cùng thuộc lớp đó, nó truyền dữ liệu qua cho đối tượng đang xét để tạo lập nên đối tượng đó.

Bản chất là C++ đã cung cấp sẵn cho ta phương thức khởi tạo sao chép và ta có thể sử dụng bình thường không cần phải tạo ra lại 1 phương thức khác, nhưng tùy trường hợp mà ta phải chủ động tạo ra phương thức khởi tạo sao chép chứ không thể sử dụng của hệ thống có sẵn
(Lưu ý số 2)

1. PHƯƠNG THỨC KHỞI TẠO (CONSTRUCTOR)


11

PHƯƠNG THỨC KHỞI TẠO SAO CHÉP (COPY CONSTRUCTOR)

File: HocSinh.h

```
#pragma once
#include <iostream>
#include <string>
using namespace std;

class HocSinh
{
private:
    string HoTen;
    float DiemThi;
public:
    HocSinh(void);
    HocSinh(string, float);
    HocSinh(const HocSinh &);
    ~HocSinh(void);
};
```



Constructor - Destructor - Getter - Setter

File: HocSinh.cpp


```
#include "HocSinh.h"

HocSinh::HocSinh(void)
{
    HoTen = "Son dep trai";
    DiemThi = 9.9;
}

HocSinh::HocSinh(string hoten, float diem)
{
    HoTen = hoten;
    DiemThi = diem;
}

HocSinh::HocSinh(const HocSinh &hs)
{
    HoTen = hs.HoTen;
    DiemThi = hs.DiemThi;
}

HocSinh::~HocSinh(void)
{
}
```



2. PHƯƠNG THỨC PHÁ HỦY (DESTRUCTOR)

12

SỰ CẦN THIẾT CỦA DESTRUCTOR ?

- Giúp dọn dẹp chương trình sau khi đối tượng đã thực thi xong.
- Công việc chủ yếu là thu hồi tất cả các tài nguyên đã cấp phát cho đối tượng trong giai đoạn lúc đối tượng còn đang thực thi.
- **Lưu ý số 3**

2. PHƯƠNG THỨC PHÁ HỦY (DESTRUCTOR)

13

ĐẶC ĐIỂM CỦA PHƯƠNG THỨC PHÁ HỦY

1. Tên phương thức trùng với tên lớp và có dấu ~ đặt ở ngay phía trước.
2. Không có tham số đầu vào và cũng không có giá trị trả về.
3. Chỉ có duy nhất 1 phương thức phá hủy ở 1 lớp.
4. Ngầm định tự động chạy sau khi đối tượng hết phạm vi sử dụng.

=> Phương thức phá hủy chỉ chạy duy nhất 1 lần trong quá trình sống của 1 đối tượng.

2. PHƯƠNG THỨC PHÁ HỦY (DESTRUCTOR)

14

File: HocSinh.h

```
#pragma once
#include <iostream>
#include <string>
using namespace std;

class HocSinh
{
private:
    string HoTen;
    float DiemThi;
public:
    HocSinh(void);
    HocSinh(string, float);
    HocSinh(const HocSinh &);
    ~HocSinh(void);
};
```

File: HocSinh.cpp

```
#include "HocSinh.h"

HocSinh::HocSinh(void)
{
    HoTen = "Son dep trai";
    DiemThi = 9.9;
}

HocSinh::HocSinh(string hoten, float diem)
{
    HoTen = hoten;
    DiemThi = diem;
}

HocSinh::HocSinh(const HocSinh &hs)
{
    HoTen = hs.HoTen;
    DiemThi = hs.DiemThi;
}

HocSinh::~HocSinh(void)
{
    /* Để trống nha do dữ liệu đầu vào
    của ta không phải là con trỏ, còn nếu như là con trỏ
    thì ta phải giải phóng bộ nhớ cho nó trong phương thức này */
}
```

3. GIẢI ĐÁP NHỮNG LƯU Ý

15

LƯU Ý SỐ 1

Khi ta khởi tạo 1 đối tượng thì phương thức thiết lập mặc định của đối tượng đó tự động được gọi lên do trình biên dịch hỗ trợ (mặc dù ta không cần khai báo), nhưng nếu trong bài ta có sử dụng đến các phương thức thiết lập khác như phương thức thiết lập nhận tham số đầu vào, phương thức thiết lập sao chép thì bắt buộc ta phải khai báo ra phương thức thiết lập mặc định vì lúc này trình biên dịch sẽ không hỗ trợ nữa, nếu không ta sẽ bị lỗi chương trình, cụ thể là: "no appropriate default constructor available".

Constructor - Destructor - Getter - Setter

3. GIẢI ĐÁP NHỮNG LƯU Ý

16

LƯU Ý SỐ 2

Nếu kiểu dữ liệu của các thành phần dữ liệu (thuộc tính) bên trong 1 lớp đối tượng không phải là kiểu dữ liệu con trỏ thì ta không cần phải cài đặt phương thức tạo lập sao chép cho lớp (C++ đã cung cấp sẵn cho ta phương thức tạo lập sao chép mặc định), tuy nhiên nếu đề bài yêu cầu thì cứ khai báo cũng chả chết. Còn nếu 1 lớp đối tượng có thuộc tính là con trỏ thì bắt buộc ta phải khai báo phương thức thiết lập sao chép cho lớp đó chứ không thể dùng phương thức tạo lập sao chép mặc định do C++ cung cấp (vì nó chỉ sao chép địa chỉ của con trỏ chứ không thật sự sao chép vùng nhớ mà chúng quản lý - dẫn đến 2 đối tượng cùng trỏ đến 1 địa chỉ vùng nhớ, nếu 1 trong 2 đối tượng thay đổi thì cả 2 đều bị thay đổi theo => bị sai).

Constructor - Destructor - Getter - Setter

3. GIẢI ĐÁP NHỮNG LƯU Ý

17

LƯU Ý SỐ 3

Phương thức phá hủy tự động được gọi khi đối tượng hết phạm vi sử dụng, mặc định đối với những đối tượng không có thuộc tính là con trỏ thì không cần cài đặt cho phương thức phá hủy (để rỗng), còn các đối tượng có dùng con trỏ thì phải giải phóng vùng nhớ bên trong phương thức phá hủy.

4. GETTER & SETTER

18

GETTER

Cho phép lấy dữ liệu hiện tại (thuộc tính) đang ở tầm vực private của đối tượng đem ra bên ngoài cho 1 số trường hợp cần thiết. Lưu ý là chỉ lấy ra để so sánh, xem chứ không thay đổi lại được.

4. GETTER & SETTER

19

SETTER

Cho phép từ bên ngoài lớp ta có thể truy xuất vào được thuộc tính nằm bên trong lớp đang bị giới hạn bởi tầm vực là private và sẽ thay đổi lại nó nếu ta muốn (Yên tâm là ta có sự kiểm soát rõ ràng).

5. BÀI TẬP ỨNG DỤNG TẠI LỚP

20

Xây dựng lớp điểm trong không gian 2 chiều cho phép thực hiện các chức năng:

- Nhập, xuất.
- Hàm dựng: hàm dựng không tham số và hàm dựng 2 tham số.
- Lấy tọa độ.
- Gán giá trị tọa độ.
- Tính khoảng cách với một điểm khác.

5. BÀI TẬP ỨNG DỤNG TẠI LỚP

21

Viết code sao cho chạy được các hàm main dưới đây:

Bài 01

```
void main()
{
    PhanSo a;           // 0/1
    a.Xuat();
    PhanSo b(1,2);      // 1/2
    b.Xuat();
    PhanSo c(3,4);      // 3/4
    c.Xuat();
    PhanSo d(3);         // 3/1
    d.Xuat();
    PhanSo e(-5);        // -5/1
    e.Xuat();
    PhanSo f(c);         // 3/4
    f.Xuat();
    a=b.Cong(c);
    a.Xuat();
}
```

2.6 Bài 06

```
void main()
{
    // MSSV: ""; Ho ten: "", Ngay sinh: 1/1/1990, DLT: 10, DTH: 10
    SinhVien a;
    a.Xuat();
    SinhVien b("1264132");
    b.Xuat();
    SinhVien c("1264132", "Nguyen Truong An");
    c.Xuat();
    // DLT: 7, DTH: 8
    SinhVien d("1264132", "Nguyen Truong An", 7, 8);
    d.Xuat();
    Ngay ngaySinh(1990,2,3);
    SinhVien e("1264132", "Nguyen Truong An", ngaySinh);
    e.Xuat();
    SinhVien f("1264132", "Nguyen Truong An", ngaySinh, 7, 8);
    f.Xuat();
    SinhVien g("1264132", "Nguyen Truong An", 1990, 2, 3, 7, 8);
    g.Xuat();
    SinhVien h(d);
    h.Xuat();
}
```

6. BÀI TẬP VỀ NHÀ

22

Viết code sao cho chạy được các hàm main dưới đây:

Bài 02

```
void main()
{
    SoPhuc a;           // 0+0i
    a.Xuat();
    SoPhuc b(2);        // 2+0i
    b.Xuat();
    SoPhuc c(3,4);      // 3+4i
    c.Xuat();
    SoPhuc d(c);         // 3+4i
    d.Xuat();
    SoPhuc e(-5,6);     // -5+6i
    e.Xuat();
    SoPhuc f(-7);       // -7+0i
    f.Xuat();
}
```

Bài 03

```
void main()
{
    Ngay a;             // 1/1/1
    a.Xuat();
    Ngay b(2013);       // 1/1/2013
    b.Xuat();
    Ngay c(2013,5);     // 1/5/2013
    c.Xuat();
    Ngay d(2013,2,3);   // 3/2/2013
    d.Xuat();
    Ngay e(2013,1,31);  // 31/1/2013
    e.Xuat();
    Ngay f(2013,2,-8);  // 1/2/2013
    f.Xuat();
    Ngay g(b);
    g.Xuat();
    Ngay h=d;
    h.Xuat();
}
```

6. BÀI TẬP VỀ NHÀ

23

Viết code sao cho chạy được các hàm main dưới đây:

Bài 04

```
void main()
{
    ThoiGian a;           // 00:00:00
    a.Xuat();
    ThoiGian b(7);        // 07:00:00
    b.Xuat();
    ThoiGian c(7,30);     // 07:30:00
    c.Xuat();
    ThoiGian d(8,12,56);  // 08:12:56
    d.Xuat();
    ThoiGian e(b);
    e.Xuat();
    ThoiGian f=c;
    f.Xuat();
}
```

Bài 05

```
void main()
{
    DuongTron a;          // 0,0-1
    a.Xuat();
    Diem d1(-7,8);
    DuongTron b(d1);      // -7,8-1
    b.Xuat();
    DuongTron c(10);      // 0,0-10
    c.Xuat();
    DuongTron d(-7,8,10); // -7,8-10
    d.Xuat();
    DuongTron e(d1,12);   // -7,8-12;
    e.Xuat();
    DuongTron f(b);
    f.Xuat();
    DuongTron g=a;
    g.Xuat();
}
```