

## Bài 2: Cấu tử và Huỷ tử

---



### Nội dung

---

1. Con trỏ this
2. Cấu tử và sử dụng cấu tử
3. Huỷ tử và sử dụng huỷ tử
4. Cấu tử và huỷ tử ngầm định
5. Quá tải và quá tải cấu tử
6. Cấu tử sao chép
7. Ôn lại
8. Bài tập

## Con trỏ this

- C++ đã ngầm định sử dụng một con trỏ đặc biệt với tên gọi **this**.
- Các thuộc tính viết trong phương thức được hiểu là thuộc một đối tượng do con trỏ this trỏ tới.
- Ví dụ lớp Hinhtron
- ```
void Hinhtron::DatBankinh(int bk)
{
    bankinh = bk;
}
```
- Ta có thể viết tường minh hơn với con trỏ this.  

```
void Hinhtron::DatBankinh(int bk)
{
    this -> bankinh = bk;
}
```

3

## Cấu tử - hàm tạo (Constructor)

- Cấu tử là gì?
  - Cấu tử (hàm tạo) cũng là một phương thức của lớp (nhưng là hàm đặc biệt) dùng để tạo dựng một đối tượng mới.
- Đặc điểm
  - Cấu tử là phương thức được gọi ngay khi đối tượng được tạo ra.
  - Cấu tử cho phép khởi tạo dữ liệu của đối tượng ngay khi được tạo ra.
  - Cấu tử phải có tên giống với tên lớp.
  - Không khai báo kiểu cho hàm tạo.
  - Hàm tạo không có kết quả trả về.
- Ví dụ  

```
class Hinhtron
{
private:
    int bankinh;
public:
    Hinhtron(int a);
    // Các phương thức khác...
};
```

4

## Cấu tử

- Định nghĩa nội dung cấu tử

```
Hinhtron::Hinhtron(int a)
{
    bankinh = a;
}
```

- Sử dụng

```
int main()
{
    Hinhtron x(5);
    cout<<"Diện tích = "<<x.LayDientich();
    x.DatBankinh(2);
    cout<<"Diện tích = "<<x.LayDientich();
    return 0;
}
```

Cấu tử sẽ được gọi cho đối tượng x và đưa giá trị 5 vào bán kính của x.

Đặt lại bán kính cho x.

5

## Cấu tử

- Cho biết kết quả trên màn hình nếu cấu tử của Hinhtron được định nghĩa như sau:

```
Hinhtron::Hinhtron(int a)
{
    cout<<"Xin chào, tôi là hình tron"<<endl;
    bankinh = a;
}

int main()
{
    Hinhtron x(5), y(7);
    cout<<"Diện tích của x = "<<x.LayDientich();
    x.DatBankinh(2);
    y.DatBankinh(4);
    cout<<"Diện tích x = "<<x.LayDientich();
    cout<<"Diện tích y = "<<y.LayDientich();
    return 0;
}
```

6

## Khởi tạo dữ liệu với cấu tử

### ■ Các cách khởi tạo dữ liệu bằng cấu tử

- `Hinhtron::Hinhtron(int a)`
- {
- `bankinh = a;` ← Gán dữ liệu trực tiếp
- }
- `Hinhtron::Hinhtron(int a) : bankinh(a)` ← Khởi tạo theo cách viết của OOP
- {
- }
- `Hinhtron::Hinhtron(int a)`
- {
- `DatBankinh(a);` ← Gọi phương thức Đặt (Set)
- }

7

## Hủy tử - hàm hủy(Destructor)

### ■ Hủy tử

- là một hàm thành viên của lớp (phương thức) có chức năng ngược với hàm tạo. Hủy tử được gọi trước khi đối tượng được hủy bỏ.

### ■ Đặc điểm

- là hàm không có kiểu, không có giá trị trả về.
- Tên của hủy tử gồm một dấu ngã (đứng trước) và tên lớp: `~Tên_lớp`
- Hủy tử không có đối

8

## Huỷ tử

- Ví dụ

```
class Hinhtron
{
private:
    int bankinh;
public:
    ~Hinhtron();
    // Các phương thức khác...
};
```

9


## Huỷ tử

- Màn hình sẽ có gì khi chương trình kết thúc với cấu tử/huỷ tử như dưới đây ?

```
■ Hinhtron::Hinhtron(int a)
■ {
■     cout<<"Xin chao, toi la hinh tron"<<endl;
■     bankinh = a;
■ }
■ Hinhtron::~Hinhtron()
■ {
■     cout<<"Tam biet !"<<endl;
■ }

■ int main()
■ {
■     Hinhtron x(5);
■     cout<<"Dien tich = "<<x.LayDientich();
■     return 0;
■ }
```

10



## Cấu tử và huỷ tử ngầm định

- Nếu một lớp không khai báo cấu tử/huỷ tử, nó sẽ sử dụng cấu tử/huỷ tử mặc định của chương trình.
- Cấu tử/huỷ tử mặc định không làm gì.

11



## Chú ý

- Cấu tử của một lớp có thể có các đối số tùy ý như một phương thức thông thường. Tuy nhiên nó không trả về giá trị.
  - `Hinhtron(int a);`
  - `Hinhtron();`
  - `Hinhtron(Hinhtron &ht);` // cấu tử sao chép
- Huỷ tử của của một lớp không có đối số và không trả về giá trị.
  - `~Hinhtron();`

12

## Quá tải hàm

- Quá tải hàm là việc định nghĩa nhiều hàm cùng tên nhưng có đối số khác nhau. Khi gọi hàm, chương trình sẽ kiểm tra các đối số để biết hàm nào được gọi.
- Các phương thức về bản chất là hàm nên ta có thể quá tải chúng
- *Cấu tử của một lớp nên được quá tải để cung cấp nhiều cách khởi tạo đa dạng.*
- C++ không cho phép quá tải huỷ tử.

13

## Quá tải cấu tử

- `class Hinhtron`
- `{`
- `private:`
- `int bankinh;`
- `public:`
- `// Cấu tử không đối số (khởi tạo hình tròn mặc định)`
- **`Hinhtron();`**
- `// Cấu tử có đối số (khởi tạo hình tròn với giá trị cho trước)`
- **`Hinhtron(int a);`**
- `// Cấu tử sao chép - có đối số tham chiếu tới một hình tròn`
- `// (khởi tạo hình tròn giống một hình tròn có trước)`
- **`Hinhtron(Hinhtron &x);`**
- `};`

14

## Quá tải cấu tử

```
Hinhtron::Hinhtron() : bankinh(0)
{
}
Hinhtron::Hinhtron(int a) : bankinh(a)
{
}
Hinhtron::Hinhtron(Hinhtron &ht)
{
    bankinh = ht.bankinh; // Lấy dữ liệu từ đối tượng có sẵn
};
```

15

## Sử dụng các cấu tử

```
int main()
{
    Hinhtron x, y(5), z(x);
    cout<<"Dien tich x = "<<x.LayDientich()<<endl;
    cout<<"Dien tich y = "<<y.LayDientich()<<endl;
    cout<<"Dien tich z = "<<z.LayDientich()<<endl;
    x.DatBankinh(8);
    y.DatBankinh(2);
    cout<<endl;
    cout<<"Dien tich x = "<<x.LayDientich()<<endl;
    cout<<"Dien tich y = "<<y.LayDientich()<<endl;
    cout<<"Dien tich z = "<<z.LayDientich()<<endl;
    return 0;
}
```

16



## Cấu tử sao chép

- Cấu tử sao chép khởi tạo một đối tượng mới từ đối tượng có sẵn.  
**Hinhtron(Hinhtron &x);** // Khai báo cấu tử sao chép
- Nếu một hàm có *đối số giá trị là một đối tượng* thì khi gọi hàm, chương trình sẽ tạo ra một đối tượng mới trong ngăn xếp và *cấu tử sao chép được gọi*.
- Nếu không định nghĩa cấu tử sao chép, *cấu tử sao chép mặc định* sẽ được gọi. Nó gán giá trị của đối số thực vào đối số hình thức (giống phép gán).
- Cấu tử sao chép thường được dùng với những lớp có liên quan đến cấp phát bộ nhớ động.

17

## Cấu tử sao chép

- Ví dụ (giả sử lớp Hinhtron đã được định nghĩa ở trên):

```
Hinhtron::Hinhtron(Hinhtron &htron)
{
    cout<<"Goi cau tu sao chep";
    bankinh = htron.bankinh;
}
void test(Hinhtron htron)
{
    // Không làm gì
}
int main()
{
    Hinhtron x;
    test(x); // Cấu tử sao chép được gọi để tạo ra đối tượng htron
            // trong stack.
    return 0;
}
```

18

## Ví dụ về cấu tử và huỷ tử

- Cho biết kết quả trên màn hình của chương trình sau
- `#include <iostream.h>`
- `class Wonder`
- `{`
- `public:`
- `Wonder() { cout<<"Goi cau tu khong co tham so"<<endl; }`
- `Wonder(int a) { cout<<"Goi cau tu co tham so"<<endl; }`
- `Wonder(Wonder &w) { cout<<"Goi cau tu sao chep"<<endl; }`
- `~Wonder() { cout<<"Goi Huy tu"<<endl; }`
- `};`

19

## Ví dụ về cấu tử và huỷ tử

- `void test1(Wonder a)`
- `{`
- `cout<<"Goi ham test1"<<endl;`
- `}`
- `void test2(Wonder &a)`
- `{`
- `cout<<"Goi ham test2"<<endl;`
- `}`
- `void test3()`
- `{`
- `cout<<"Goi ham test3"<<endl;`
- `Wonder m, h(3);`
- `}`

20

## Ví dụ về cấu tử và huỷ tử

```
int main()
{
    Wonder x, y(3), z(y);
    test1(x);
    test2(y);
    test3();
    cout<<"Ket thuc"<<endl;
    return 0;
}
```

21

## Ví dụ về cấu tử và huỷ tử

1. Goi cau tu khong co tham so
2. Goi cau tu co tham so
3. Goi cau tu sao chep
4. Goi cau tu sao chep
5. Goi ham test1
6. Goi Huy tu
7. Goi ham test2
8. Goi ham test3
9. Goi cau tu khong co tham so
10. Goi cau tu co tham so
11. Goi Huy tu
12. Goi Huy tu
13. Ket thuc
14. Goi Huy tu
15. Goi Huy tu
16. Goi Huy tu

22

## Cấu trúc một chương trình HĐT

- Kiểu 1: Tất cả mã lệnh trong cùng một file .cpp

```
#include <...>
```

```
class Hinhtron... // khai báo lớp
```

```
// định nghĩa các phương thức của lớp
```

```
Hinhtron::Hinhtron()...
```

```
void Hinhtron::DatBankinh(int bk)...
```

```
void main() // hàm main
```

```
{  
}
```

23

## Cấu trúc một chương trình HĐT

- Kiểu 2: Tách riêng file định nghĩa lớp và main()
- File định nghĩa lớp (giả sử là *Hinhtron.h*) chứa phần khai báo lớp và cài đặt các phương thức của lớp.
- File chứa hàm main:
  - **#include <...>**
  - **#include "Hinhtron.h"**
  - void main()
  - {
  - }

24

## Cấu trúc một chương trình HĐT

- Kiểu 3 (hay dùng): Tách mỗi lớp thành 2 file, file khai báo lớp (.h) và file cài đặt lớp (.cpp).
- Ví dụ:
  - File Hinhtron.h chứa phần khai báo lớp  
*class Hinhtron...*
  - File Hinhtron.cpp chứa phần cài đặt các phương thức  
*Hinhtron::Hinhtron()...*  
*void Hinhtron::DatBankinh(int bk)...*
  - File ctchinh.cpp chứa hàm main()  
*#include <...>*  
*#include "Hinhtron.h"*

25

## Ôn lại

1. Con trỏ this
2. Các đặc điểm của cấu tử ?
3. Các đặc điểm của huỷ tử ?
4. Một chương trình có thể không có cấu tử và huỷ tử được không ?
5. Quá tải là gì ?
6. Có thể quá tải huỷ tử được không ?
7. Tại sao nên quá tải cấu tử ?
8. Cấu tử sao chép là gì ?
9. Cấu tử sao chép được gọi khi nào ?

26



## Bài tập về nhà

1. Cài đặt lớp Hinhtron với các cấu tử và huỷ tử. Sử dụng các đối tượng hình tròn một cách sáng tạo.
2. Xây dựng lớp Hinhchunhat với các phương thức cơ bản. Thêm phương thức tính độ dài đường chéo và phương thức kiểm tra đó có phải là hình vuông không. Viết chương trình đếm xem có bao nhiêu hình chữ nhật nhập vào là hình vuông.