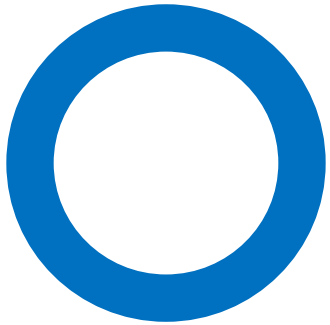
Hàm dựng và Hàm hủy

Lập trình hướng đối tượng

Hồ Tuấn Thanh

htthanh@fit.hcmus.edu.vn

Hàm dựng



class PhanSo

```
class PhanSo
{
private:
    int tu, mau;
public:
    void xuat();
};
```

```
void PhanSo::xuat()
{
    if(mau < 0)
    {
        tu = -tu;
        mau = -mau;
    }
    cout << tu << "/" << mau;
}
```

Giá trị mặc định của các thuộc tính

```
PhanSo p1;
p1.xuat();
```

- Giá trị in ra màn hình là bao nhiêu?
- Lỗi hoặc giá trị không xác định (rác)
- Làm thế nào để thiết lập giá trị cụ thể của phân số khi vừa được tạo ra.

Giảm pháp tạm thời ➔ Hàm init()

```
void PhanSo::init(int t, int m)
{
    if(m == 0)
        m = 1;
    tu = t;
    mau = m;
}
```

```
PhanSo p1;
p1.init(3, 4);
p1.xuat();
```

Nếu quên gọi hàm init()

- Nếu developer khi sử dụng class PhanSo, quên gọi hàm init thì sao?

```
void PhanSo::init(int t, int m)
{
    if(m == 0)
        m = 1;
    tu = t;
    mau = m;
}
```

```
PhanSo p1;
p1.init(3, 4);
p1.xuat();
```

Giải pháp tốt hơn ➔ Hàm dựng

```
class PhanSo
{
private:
    int tu, mau;
public:
    void xuat();
    PhanSo(int t, int m);
};
```

```
PhanSo::PhanSo(int t, int m)
{
    if(m == 0)
        m = 1;
    tu = t;
    mau = m;
}
```

```
PhanSo p1(3, 4);
p1.xuat();
```

Giải thích

- Hàm dựng (phương thức thiết lập, constructor) là hàm được gọi **ngay sau khi đối tượng được tạo ra**.
- Khi khai báo:

```
PhanSo p1(3, 4);
```

- Đối tượng phân số p1 được tạo ra.
- Hàm dựng 2 tham số của class PhanSo được kích hoạt

```
PhanSo::PhanSo(int t, int m)
{
    if(m == 0)
        m = 1;
    tu = t;
    mau = m;
}
```

Khái niệm hàm dựng

- Hàm dựng được ngay sau khi đối tượng được tạo ra
- Mỗi class có thể có nhiều hàm dựng
- Chỉ được gọi khi đối tượng được tạo ra

```
p1.PhanSo(3, 4); // ERROR!!!
```

- C++:
  - Có tên trùng với tên class
  - Không có kiểu trả về (no return value)

Hàm dựng 1 tham số

```
PhanSo p2(7); // 7/1
p2.xuat();
```

- Cần cài đặt hàm dựng tương ứng: 1 tham số, kiểu int

```
PhanSo::PhanSo(int x)
{
    tu = x;
    mau = 1;
}
```

```
// Một cách sử dụng khác
PhanSo p3 = 7;
p3.xuat();
```

Hàm dựng 1 tham số

- Cần cài đặt hàm dựng tương ứng: 1 tham số, kiểu int

```
PhanSo::PhanSo(int x)
{
    tu = x;
    mau = 1;
}
```

```
// Một cách sử dụng khác
PhanSo p3 = 7;
p3.xuat();
```

Hàm dựng mặc định & operator =

```
PhanSo p7;
p7 = 18; // p7 = PhanSo(18);
p7.xuat();
cout << endl;
```

Hàm dựng mặc định

```
PhanSo p4; // 0/1
p4.xuat();
```

Hàm dựng mặc định

- Cần cài đặt hàm dựng 0 tham số
- Còn gọi là hàm dựng mặc định, default constructor

```
PhanSo::PhanSo()
{
    tu = 0;
    mau = 1;
}
```

Hàm dựng mặc định

```
PhanSo p5();
p5.xuat();
```

- Error, compiler sẽ hiểu đây khai báo:
  - Hàm p5()
  - Không tham số
  - Trả về PhanSo

Hàm định mặc định

```
class PhanSo
{
private:
    int tu, mau;
public:
    void xuat();
};
```

```
// Ko báo lỗi biên dịch
PhanSo p4;
p4.xuat();
```

- Tại sao tuần trước, em chưa code hàm dựng nào hết, mà chỗ `PhanSo p4;` ko báo lỗi thiếu hàm dựng?

Hàm dựng mặc định

- Nếu 1 class chưa có hàm dựng nào hết, thì compiler sẽ tạo sẵn 1 hàm dựng mặc định cho class đó.

Hàm dựng mặc định

```
class PhanSo
{
private:
    int tu, mau;
public:
    void xuat();
    PhanSo(int t, int m);
    PhanSo(int x);
};
```

```
PhanSo::PhanSo(int t, int m)
{
    if(m == 0)
        m = 1;
    tu = t;
    mau = m;
}
```

```
PhanSo::PhanSo(int x)
{
    tu = x;
    mau = 1;
}
```

```
PhanSo p1; // Error!!!
p1.xuat();
PhanSo p2(7);
p2.xuat();
PhanSo p3(5, 6);
p3.xuat();
```

- Tại sao `PhanSo p1;` lỗi???

Hàm dựng mặc định

- Khi 1 class đã có một số hàm dựng rồi, thì class đó phải tự cài hàm dựng mặc định, compiler sẽ ko cho nữa.

Hàm dựng sao chép

```
PhanSo p1(3, 4);
p1.xuat();
PhanSo p6(p1); // 3/4
p6.xuat();
PhanSo p7 = p1; // 3/4
p7.xuat();
```

Hàm dựng sao chép

- Chỗ này, p6 và p7 sẽ gọi hàm dựng sao chép (copy constructor), sao chép giá trị tử mẫu của p1 sang p6 và p7.
- Trong C++, ta không cần cài đặt hàm dựng sao chép, vì compiler đã cho ta cài đặt hàm dựng sao chép mặc định.

- Nếu thuộc tính là con trỏ, thì cần cài đặt hàm dựng sao chép. (next week!!!)

Hàm dựng sao chép

- Nếu muốn cài đặt hàm dựng sao chép, thì đây là code.

```
PhanSo::PhanSo(const PhanSo &other)
{
    tu = other.tu;
    mau = other.mau;
}
```

Hàm dựng sao chép

```
#include<iostream>
using namespace std;

class PhanSo
{
private:
    int tu, mau;
public:
    void xuat();
    void init(int t, int m);
    PhanSo(int t, int m);
    PhanSo(int x);
    PhanSo();
    PhanSo(const PhanSo &other);
    PhanSo cong(PhanSo b);
};
```

Hàm dựng sao chép

```
void PhanSo::xuat()
{
    if(mau < 0)
    {
        tu = -tu;
        mau = -mau;
    }
    cout << tu << "/" << mau;
}
```

```
void PhanSo::init(int t, int m)
{
    if(m == 0)
        m = 1;
    tu = t;
    mau = m;
}
```

Hàm dựng sao chép

```
PhanSo::PhanSo(int t, int m)
{
    cout << "2-param cons" << endl;
    if(m == 0)
        m = 1;
    tu = t;
    mau = m;
}
```

```
PhanSo::PhanSo(int x)
{
    cout << "1-param cons" << endl;
    tu = x;
    mau = 1;
}
```

```
PhanSo::PhanSo()
{
    cout << "default cons" << endl;
    tu = 0;
    mau = 1;
}
```

```
PhanSo::PhanSo(const PhanSo &other)
{
    cout << "copy cons" << endl;
    tu = other.tu;
    mau = other.mau;
}
```

Hàm dựng sao chép

```
PhanSo PhanSo::cong(PhanSo b)
{
    cout << "cong 2 ps" << endl;
    PhanSo kq;
    kq.tu = tu*b.mau + mau*b.tu;
    kq.mau = mau * b.mau;
    return kq;
}
```

Hàm dựng sao chép

```
int main()
{
    PhanSo p1(1,2);
    PhanSo p2(3,4);
    PhanSo p3 = p1.cong(p2);
    p3.xuat();
    cout << endl;
    return 0;
}
```

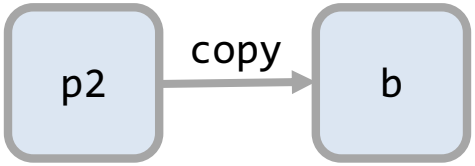
```
2-param cons
2-param cons
copy cons
cong 2 ps
default cons
copy cons
copy cons
10/8
```

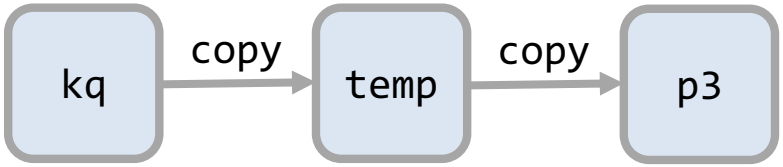
Hàm dựng sao chép

- Hàm dựng sao chép được gọi khi pass by value và khi return by value.

```
PhanSo PhanSo::cong(PhanSo b)
{
    cout << "cong 2 ps" << endl;
    PhanSo kq;
    kq.tu = tu*b.mau + mau*b.tu;
    kq.mau = mau * b.mau;
    return kq;
}
```

```
int main()
{
    PhanSo p1(1,2);
    PhanSo p2(3,4);
    PhanSo p3 = p1.cong(p2);
    p3.xuat();
    cout << endl;
    return 0;
}
```





Hàm dựng sao chép

```
PhanSo PhanSo::cong(PhanSo &b)
{
    cout << "cong 2 ps" << endl;
    PhanSo kq;
    kq.tu = tu*b.mau + mau*b.tu;
    kq.mau = mau * b.mau;
    return kq;
}
```

```
int main()
{
    PhanSo p1(1,2);
    PhanSo p2(3,4);
    PhanSo p3 = p1.cong(p2);
    p3.xuat();
    cout << endl;
    return 0;
}
```

```
2-param cons
2-param cons
cong 2 ps
default cons
copy cons
copy cons
10/8
```

Hàm dựng sao chép

```
PhanSo& PhanSo::cong(PhanSo &b)
{
    cout << "cong 2 ps" << endl;
    PhanSo kq;
    kq.tu = tu*b.mau + mau*b.tu;
    kq.mau = mau * b.mau;
    return kq;
}
```

```
int main()
{
    PhanSo p1(1,2);
    PhanSo p2(3,4);
    PhanSo p3 = p1.cong(p2);
    p3.xuat();
    cout << endl;
    return 0;
}
```

```
2-param cons
2-param cons
cong 2 ps
default cons
10/8
```

```
cc1plus: warnings being treated as errors
t.cpp: In member function 'PhanSo& PhanSo::cong(PhanSo)':
warning: reference to local variable 'kq' returned
```

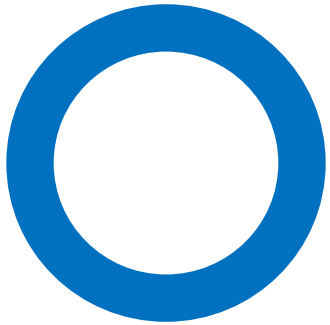
Hàm dựng & con trỏ

```
// ko gọi hàm dựng nào hết vì p17 chưa trỏ đến phân số nào cả
PhanSo *p17;
// ko gọi hàm dựng nào hết vì ko có phân số MỚI nào được tạo ra cả
p17 = &p10;
// gọi hàm dựng mặc định
p17 = new PhanSo;
// gọi hàm dựng 1 tham số
p17 = new PhanSo(13);
// gọi hàm dựng sao chép
p17 = new PhanSo(p10);
// gọi hàm dựng 2 tham số
p17 = new PhanSo(7, 8);
p17->xuat();
```

Hàm dựng 2 tham số

```
PhanSo p8 = {8, 10};
p8.xuat();
cout << endl;
```

Hàm hủy



Khái niệm hàm hủy

- Hàm hủy (phương thức phá hủy, destructor) được gọi ngay trước khi đối tượng được hủy.
- Mỗi class chỉ có 1 hàm hủy duy nhất.

```
PhanSo PhanSo::cong(PhanSo b)
{
    cout << "cong 2 ps" << endl;
    PhanSo kq;
    kq.tu = tu*b.mau + mau*b.tu;
    kq.mau = mau * b.mau;
    return kq;
}
```

```
PhanSo::~PhanSo()
{
    cout << "destructor" << endl;
}
```

```
int main()
{
    PhanSo p1(1,2);
    PhanSo p2(3,4);
    PhanSo p3 = p1.cong(p2);
    p3.xuat();
    cout << endl;
    return 0;
}
```

```
2-param cons
2-param cons
copy cons
cong 2 ps
default cons
copy cons
destructor
copy cons
destructor
destructor
10/8
destructor
destructor
destructor
```


Q
&
A