

Vòng đời đối tượng

GV. Nguyễn Minh Huy



- Phương thức khởi tạo.
- Phương thức hủy.
- Thành phần tĩnh.
- Class Template.



- **Phương thức khởi tạo.**
- Phương thức hủy.
- Thành phần tĩnh.
- Class Template.

Phương thức khởi tạo



■ Vấn đề khởi tạo thông tin đối tượng:

■ Giá trị ban đầu của thuộc tính?

```
class PhanSo
{
private:
    int    m_tu;
    int    m_mau;
};

void main()
{
    PhanSo p;
    // Giá trị của p??
}
```

■ Khởi tạo bằng phương thức truy xuất (getter/setter):

```
class PhanSo
{
public:
    void ganTu(int tu);
    void ganMau(int mau);
};

void main()
{
    PhanSo p;
    p.ganTu(1);
    p.ganMau(3);
}
```

**Người dùng
quên gọi?!**

Phương thức khởi tạo



■ Tính chất phương thức khởi tạo:

- “Làm khai sinh” cho đối tượng!!
- Bắt buộc gọi khi khai báo đối tượng.
- Có thể nạp chồng nhiều phương thức.
- Không có giá trị trả về.
- Có tên trùng tên lớp (trong C++).

```
class PhanSo
{
private:
    int    m_tu;
    int    m_mau;
public:
    PhanSo(int tu, int mau);
    PhanSo(int giaTri);
};
```

```
void main()
{
    PhanSo p1(1, 2);
    PhanSo p2( 5 );
    PhanSo *p3 = new PhanSo(2, 3);
}
```

Phương thức khởi tạo



- Phương thức khởi tạo mặc định:
 - “Làm khai sinh” mặc định!!
 - Không có tham số.
 - Nếu lớp không có phương thức khởi tạo nào:
→ Trình biên dịch cấp.

```
class PhanSo
{
private:
    int    m_tu;
    int    m_mau;
public:
    PhanSo();
};
```

```
void main()
{
    PhanSo p;
    PhanSo *q = new PhanSo;
}
```

Phương thức khởi tạo



■ Phương thức khởi tạo sao chép:

- “Làm khai sinh” bằng sao chép đối tượng khác.
- Tham số là đối tượng cùng lớp.
- Nếu lớp không có phương thức khởi tạo sao chép:
→ Trình biên dịch cấp.

```
class PhanSo
{
private:
    int    m_tu;
    int    m_mau;
public:
    PhanSo(const PhanSo &p);
};
```

```
void main()
{
    PhanSo p1(1, 2);

    // Sao chép p1...
    PhanSo p2(p1);
    // Sao chép p2...
    PhanSo p3 = p2;
}
```

Phương thức khởi tạo



■ Dr. Guru khuyên:

■ Một lớp nên có tối thiểu 3 phương thức khởi tạo:

- Khởi tạo mặc định.
- Khởi tạo sao chép.
- Khởi tạo với đầy đủ thông tin.

```
class PhanSo
{
private:
    int    m_tu;
    int    m_mau;
public:
    PhanSo();
    PhanSo(const PhanSo &p);
    PhanSo(int tu, int mau);
};
```





- Phương thức khởi tạo.
- **Phương thức hủy.**
- Thành phần tĩnh.
- Class Template.

Phương thức hủy



■ Vấn đề rò rỉ bộ nhớ (memory leak):

- Bộ nhớ cấp cho con trỏ không tự thu hồi.

```
class HocSinh
{
private:
    char    *m_hoTen;
};
```

```
void main()
{
    HocSinh  hs;
}
// hs.m_hoTen được thu hồi?
```

- Xây dựng phương thức thu hồi:

```
class HocSinh
{
private:
    char    *m_hoTen;
public:
    void thuHoiBoNho() {
        delete [ ]m_hoTen; }
};
```

```
void main()
{
    HocSinh  hs;
    hs.thuHoiBoNho();
}
```

**Người dùng
quên gọi?!**



■ Tính chất phương thức hủy:

- “Làm di chúc” cho đối tượng.
- Tự động gọi khi đối tượng bị hủy.
- Duy nhất cho mỗi lớp.
- Có tên **~<Tên lớp>** (trong C++).

```
class HocSinh
{
private:
    char    *m_hoTen;
    float   m_diemVan;
    float   m_diemToan;
public:
    ~HocSinh() { delete [ ]m_hoTen; }
};

void main()
{
    HocSinh  hs;
    HocSinh  *p = new HocSinh;
    delete p;
    // p->m_hoTen được hủy.
}
// hs.m_hoTen được hủy.
```



- Phương thức khởi tạo.
- Phương thức hủy.
- **Thành phần tĩnh.**
- Class Template.

Thành phần tĩnh



■ Chia sẻ giữa các đối tượng cùng lớp:

■ Mỗi đối tượng có bản sao riêng:

- Thuộc tính.
- Phương thức.

➔ Thành phần của đối tượng (object members).

■ Muốn dùng chung thông tin?

➔ Thành phần tĩnh (static members).

PhanSo

- Tử số
- Mẫu số
- Rút gọn()

p1: PhanSo

- Tử số: 1
- Mẫu số: 2
- Rút gọn()

p2: PhanSo

- Tử số: 1
- Mẫu số: 3
- Rút gọn()



■ Tính chất thành phần tĩnh:

- Thuộc tính, phương thức thuộc phạm vi lớp.
- Dùng chung cho mọi đối tượng của lớp.
- Các sử dụng (trong C++):
 - Khai báo: từ khóa “**static**”.
 - Khởi tạo: bên ngoài lớp.
 - Truy xuất: **tên lớp** kèm toán tử **::**.

```
class PhanSo
{
private:
    static int m_giaTriLN;
public:
    static int layGiaTriLN();
};
```

```
int PhanSo::m_giaTriLN = 10000;

void main()
{
    int x = PhanSo::layGiaTriLN();
}
```



- Phương thức khởi tạo.
- Phương thức hủy.
- Thành phần tĩnh.
- **Class Template.**



■ Xét lớp mảng:

- Các phần tử là số nguyên.
- Tổng quát: các phần tử kiểu bất kỳ.
 - ➔ Tham số hóa thuộc tính, phương thức.
 - ➔ Class Template.

Class Template



■ Cách sử dụng Class Template:

```
template <class T>
class Mang
{
private:
    int      m_kichThuoc;
    T        *m_duLieu;
public:
    Mang( int kichThuoc );
    T& layPhanTu( int i );
    T timMax( );
};
```

```
void main()
{
    Mang<int>      m1( 10 );
    int  a = m1.layPhanTu(5);
    int  max1 = m1.timMax( );

    Mang<PhanSo> m2( 5 );
    PhanSo  p = m2.layPhanTu( 2 );
    PhanSo  max2 = m2.timMax( );
}
```



■ Phương thức khởi tạo:

- “Làm khai sinh” cho đối tượng.
- Bắt buộc gọi khi khai báo.
- Có thể nạp chồng.

■ Phương thức hủy:

- “Làm di chúc” cho đối tượng.
- Tự động gọi khi hủy.
- Có duy nhất một.



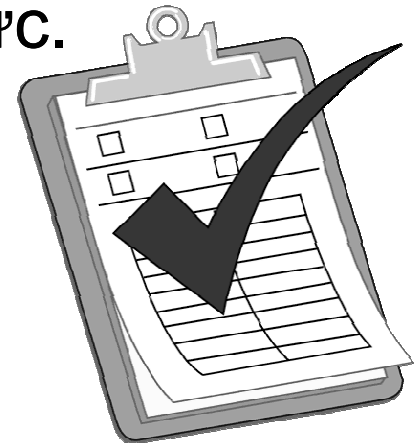


■ Thành phần tĩnh:

- Dùng chung cho đối tượng của lớp.
- Cách dùng (C++):
 - Khai báo bằng từ khóa “static”.
 - Khởi tạo bên ngoài lớp.
 - Truy xuất bằng toán tử ::.

■ Class Template:

- Tham số hóa kiểu thuộc tính, phương thức.
- Lưu trữ và xử lý tổng quát trong lớp.

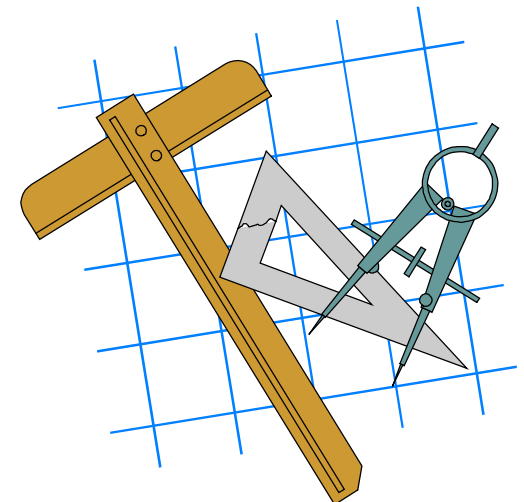




■ Bài tập 3.1:

Trang bị cho lớp **phân số** những cách khởi tạo sau:

- Khởi tạo mặc định phân số = 0.
- Khởi tạo với tử và mẫu cho trước.
- Khởi tạo từ giá trị nguyên cho trước.
- Khởi tạo từ một phân số khác.

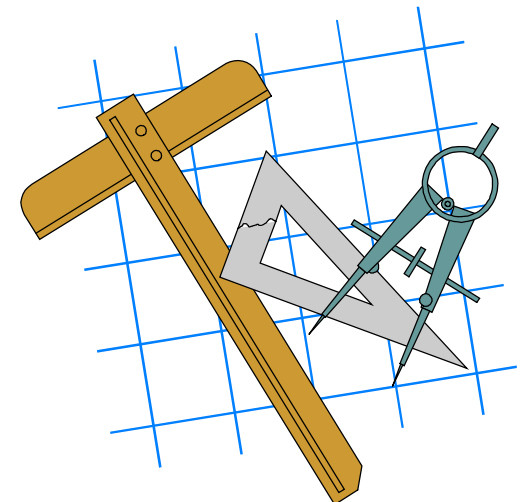




■ Bài tập 3.2:

Trang bị cho lớp **học sinh** những cách khởi tạo và hủy sau:

- Khởi tạo với họ tên và điểm văn, toán cho trước.
- Khởi tạo với họ tên cho trước, điểm văn, toán = 0.
- Khởi tạo từ một học sinh khác.
- Hủy đối tượng học sinh, thu hồi bộ nhớ.

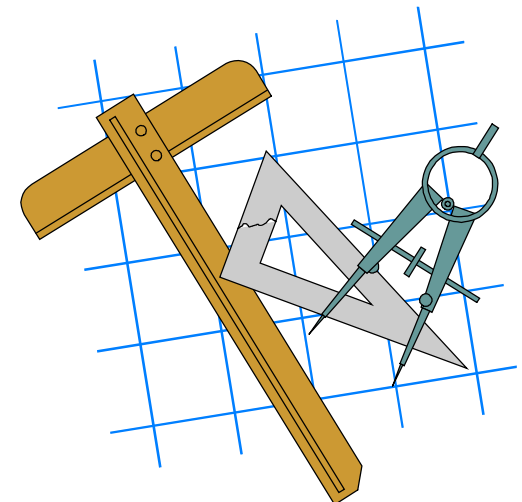




■ Bài tập 3.3:

Trang bị cho lớp **mảng** số nguyên những cách khởi tạo và hủy sau:

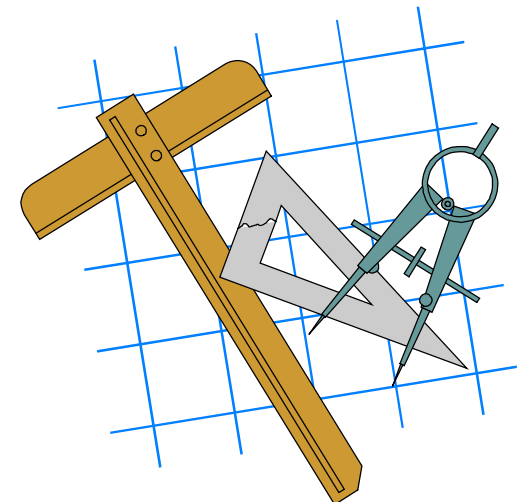
- Khởi tạo mặc định mảng kích thước = 0.
- Khởi tạo với kích thước cho trước, các phần tử = 0.
- Khởi tạo từ một mảng `int []` với kích thước cho trước.
- Khởi tạo từ một đối tượng mảng khác.
- Hủy đối tượng mảng, thu hồi bộ nhớ.





■ Bài tập 3.4:

Làm lại bài 3.3 với lớp **mảng** có phần tử thuộc một kiểu bất kỳ.





■ Bài tập 3.5 (*):

Trang bị cho lớp **phân số** những khả năng sau:

- a) Đếm tổng số lượng phân số được tạo ra trong hàm main().
- b) Hạn chế duy nhất một đối tượng phân số được tạo ra.

