Hàm dựng và Hàm hủy

Lập trình hướng đối tượng

Hồ Tuấn Thanh

htthanh@fit.hcmus.edu.vn

Hàm hủy

- Trong lớp MangSoNguyen, ta sử dụng con trỏ a để chứa một danh sách các số nguyên.
- Để tránh rò rỉ bộ nhớ, trong hàm hủy lớp MangSoNguyen, ta tiến hành kiểm tra con trỏ a và gọi toán tử delete.

```
class MangSoNguyen
{
private:
    int *a;
    int n;
public:
    MangSoNguyen(void);
    ~MangSoNguyen(void);
};
```

```
MangSoNguyen::~MangSoNguyen(void)
{
    if(a!=NULL)
    {
        delete []a;
        a=NULL;
    }
}
```

Hàm nhập

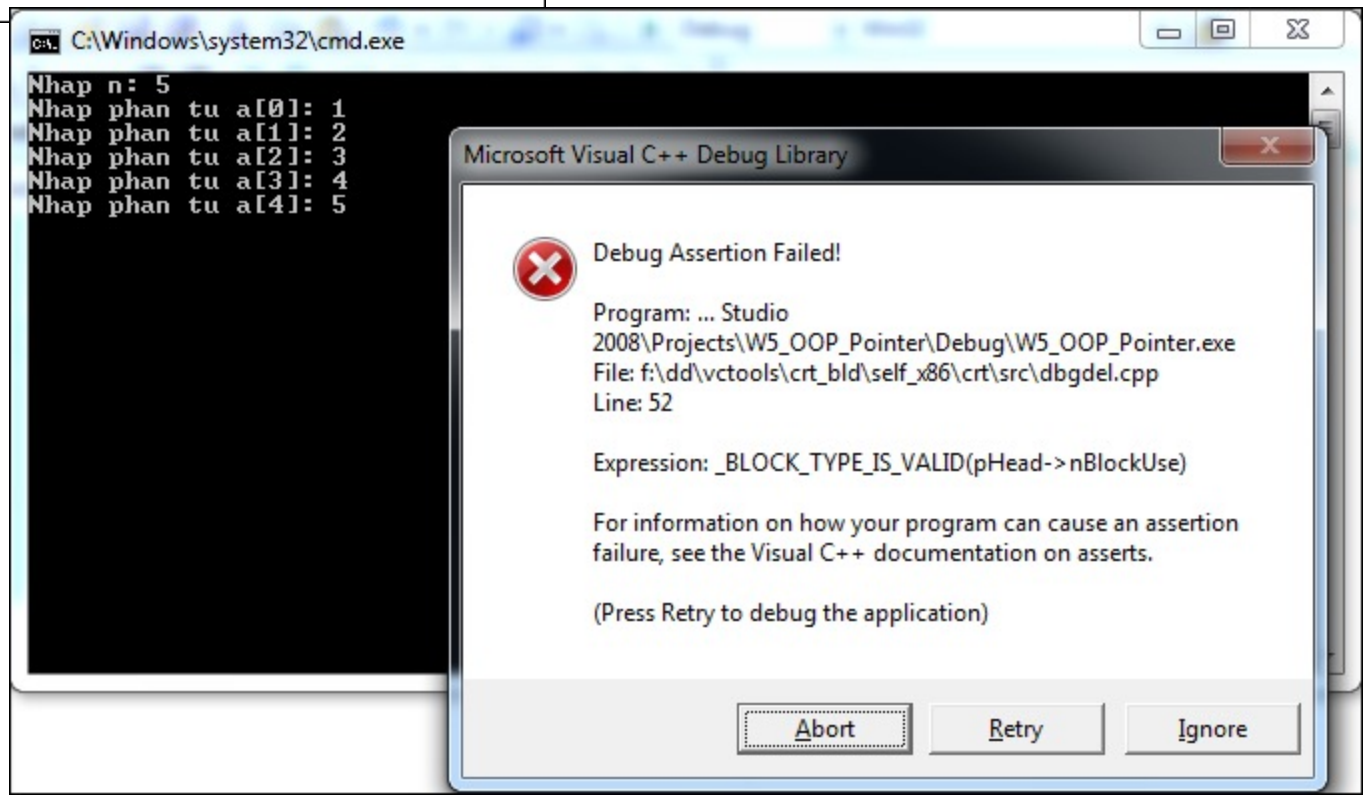
- Giả sử ta có thêm hàm Nhap trong lớp MangSoNguyen

```
class MangSoNguyen
{
private:
     int *a;
     int n;
public:
     MangSoNguyen(void);
     ~MangSoNguyen(void);
     void Nhap();
};
```

```
void MangSoNguyen::Nhap()
{
     cout<<"Nhap n: ";
     cin>>n;
     a=new int[n];
     for(int i=0;i<n;i++)
     {
          cout<<"Nhap phan tu a["<<i<<"]: ";
          cin>>a[i];
     }
}
```

Gọi hàm dựng sao chép

```
void main()
{
    MangSoNguyen m1;
    m1.Nhap();
    MangSoNguyen m2=m1;
}
```

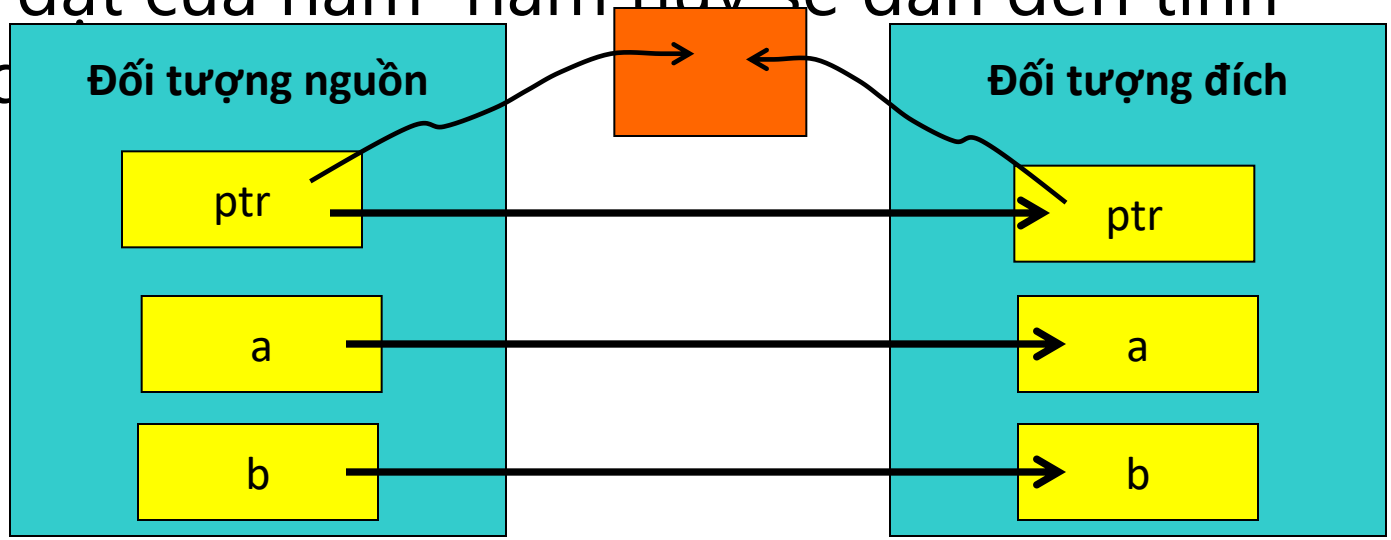


Hàm dựng sao chép mặc định

- Nếu ta không khai báo hàm dựng sao chép, thì trong mỗi lớp sẽ có sẵn một ***hàm dựng sao chép mặc định***.
- ***Hàm dựng sao chép mặc định*** thực hiện sao chép từng bit các thành phần từ đối tượng nguồn sang đối tượng đích.
- ***Hàm dựng sao chép mặc định*** làm việc rất tốt với các lớp chứa các thuộc tính thuộc kiểu dữ liệu cơ bản (int, float, double, long, PhanSo, Diem, ...)

Hàm dựng sao chép mặc định

- Tuy nhiên, vấn đề sẽ xảy ra khi tồn tại thuộc tính có kiểu con trỏ.
- Ở đây, sau khi ***sao chép mặc định***, 2 con trỏ ptr của nguồn và đích cùng trỏ về 1 vùng nhớ màu cam.
- Do cài đặt của hàm hàm hủy sẽ dẫn đến tình trạng c



Cài đặt hàm dựng sao chép

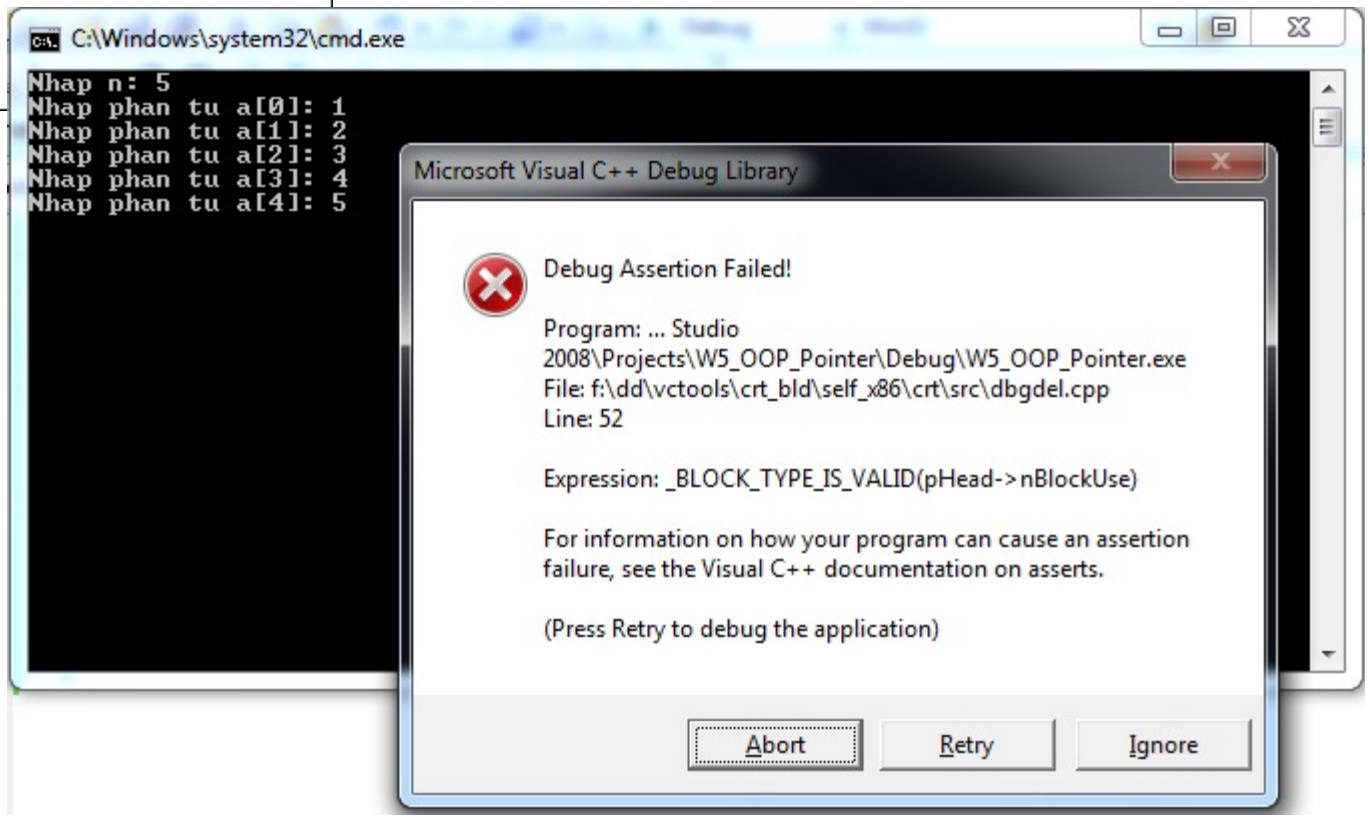
```
MangSoNguyen::MangSoNguyen(const MangSoNguyen &m)
{
    if(m.a==NULL)
    {
        a=NULL;
        n=0;
    }
    else
    {
        n=m.n;
        a=new int[n];
        for(int i=0;i<n;i++)
        {
            a[i]=m.a[i];
        }
    }
}
```

Toán tử gán bằng mặc định

- Tương tự như hàm dựng sao chép, trong một lớp, nếu không cài đặt toán tử gán bằng, lớp sẽ có sẵn ***toán tử gán bằng mặc định***.
- ***Toán tử gán bằng mặc định*** làm việc khá tốt với những kiểu dữ liệu cơ bản.
- ***Toán tử gán bằng mặc định*** sẽ phát sinh vấn đề nếu thuộc tính là con trỏ.

Toán tử gán bằng mặc định

```
void main()
{
    MangSoNguyen m1;
    m1.Nhap();
    MangSoNguyen m2=m1;
    MangSoNguyen m3;
    m3=m1;
}
```



Cài đặt toán tử gán bằng

```
MangSoNguyen& MangSoNguyen::operator=(const MangSoNguyen &m)
{
    if(this==&m)
        return *this;
    if(m.a==NULL)
    {
        a=NULL;
        n=0;
    }
    else
    {
        n=m.n;
        a=new int[n];
        for(int i=0;i<n;i++)
        {
            a[i]=m.a[i];
        }
    }
    return *this;
}
```


Q
&
A