LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Chương 4: Khởi tạo đối tượng, hàm bạn, lớp bạn (p1)

Trình bày: ThS. Lê Thanh Trọng

1. Đối tượng là thành phần của lớp
2. Đối tượng là thành phần của mảng
3. Cấp phát động đối tượng

1. Đối tượng là thành phần của lớp
2. Đối tượng là thành phần của mảng
3. Cấp phát động đối tượng

Đối tượng là thành phần của lớp

- Đối tượng có thể là thành phần của đối tượng khác, khi một đối tượng thuộc lớp "lớn" được tạo ra, các thành phần của nó cũng được tạo ra
- Ví dụ:
  - TamGiac có 3 điểm A, B, C thuộc lớp Diem
  - SinhVien có thuộc tính NgaySinh thuộc lớp MyDate

Đối tượng là thành phần của lớp

- Phương thức thiết lập (nếu có) sẽ được tự động gọi cho các đối tượng thành phần
- Khi đối tượng kết hợp bị hủy → đối tượng thành phần của nó cũng bị hủy, nghĩa là phương thức hủy bỏ sẽ được gọi cho các đối tượng thành phần, sau khi phương thức hủy bỏ của đối tượng kết hợp được gọi

Đối tượng là thành phần của lớp

```
class Diem{
    double x, y;
public:
    Diem (double xx, double yy) { x = xx; y = yy; }        //Không có khởi tạo mặc định
    // ...
};
class TamGiac{
    Diem A, B, C;
public:
    void Ve( );
    // ...
};
TamGiac t;          //Error ?
Diem d;             //Error ?
```

Đối tượng là thành phần của lớp

- Nếu đối tượng thành phần phải cung cấp tham số khi thiết lập thì đối tượng kết hợp (đối tượng lớn) phải có phương thức thiết lập để cung cấp tham số thiết lập cho các đối tượng thành phần

- Cú pháp để khởi động đối tượng thành phần là dùng dấu hai chấm (:) theo sau bởi tên thành phần và tham số khởi động

Ví dụ khởi tạo các đối tượng thành phần

```
class TamGiac{
    Diem A, B, C;
public:
    TamGiac(double xA, double yA, double xB, double yB, double xC, double yC)
    : A(xA,yA), B(xB,yB),C(xC,yC){
            cout<<“Khoi tao tam giac”;
    }
    void Ve();
    // ...
};
TamGiac t(100,100,200,400,300,300);
```

Ví dụ khởi tạo các đối tượng thành phần

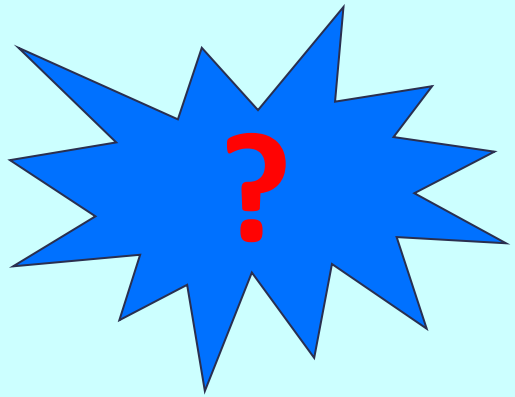
```
class TamGiac{
    Diem A,B,C;
    int loai;
public:
    TamGiac(double xA, double yA, double xB, double yB, double xC, double yC, int l):
    A(xA,yA), B(xB,yB), C(xC,yC), loai(l) {
    }
    void Ve();
    // ...
};
TamGiac t (100, 100, 200, 400, 300, 300, 1);
```

?

Cú pháp dấu hai chấm cũng được dùng cho đối tượng thành phần thuộc kiểu cơ sở

Ví dụ khởi tạo các đối tượng thành phần

```
class Diem{
    double x,y;
public:
    Diem(double xx = 0, double yy = 0) : x(xx), y(yy){
    }
    void Set(double xx, double yy){
        x = xx;
        y = yy;
    }
};
```



1. Đối tượng là thành phần của lớp
2. **Đối tượng là thành phần của mảng**
3. Cấp phát động đối tượng

Đối tượng là thành phần của mảng

- Khi một mảng được tạo ra → các phần tử của nó cũng được tạo ra → phương thức thiết lập sẽ được gọi cho từng phần tử
- Vì không thể cung cấp tham số khởi động cho tất cả các phần tử của mảng → khi khai báo mảng, mỗi đối tượng trong mảng phải có khả năng tự khởi động, nghĩa là có thể thiết lập không cần tham số → lớp phải có thể khởi tạo mặc định

Đối tượng là thành phần của mảng

- Đối tượng có khả năng tự khởi động trong những trường hợp nào?
  - Lớp không có phương thức thiết lập
  - Lớp có phương thức thiết lập không tham số
  - Lớp có phương thức thiết lập mà mọi tham số tất cả đề có giá trị mặc nhiên

Đối tượng là thành phần của mảng

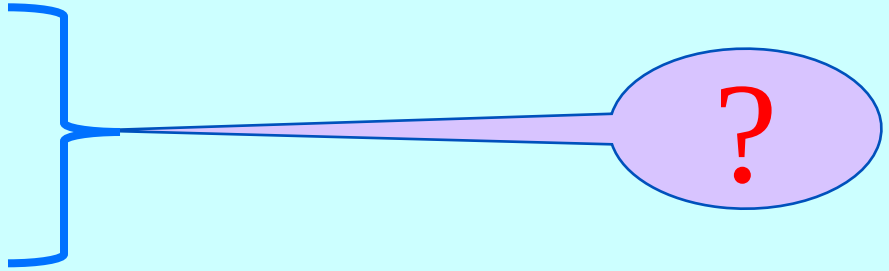
```
class Diem
{
    double x,y;
    public:
        Diem(double xx, double yy) : x(xx), y(yy) { }
        void Set(double xx, double yy) {
            x = xx, y = yy;
        }
        // ...
};
```

Đối tượng là thành phần của mảng

```
class String {
    char *p;
public:
    String(char *s) { p = strdup(s); }
    String(const String &s) { p = strdup(s.p); }
    ~String() {
        cout << "delete "<< (void *)p << "\n";
        delete [] p;
    }
};
```

Đối tượng là thành phần của mảng

```
class SinhVien{
    String MaSo;
    String HoTen;
    int NamSinh;
public:
    SinhVien(char *ht, char *ms, int ns) : HoTen(ht), MaSo(ms),
    NamSinh(ns){ }
};
String arrString[3];
Diem arrDiem[5];
SinhVien arrSV[7];
```



Thiết lập với tham số mặc nhiên

```
class Diem
{
    double x,y;
public:
    Diem(double xx = 0, double yy = 0) : x(xx), y(yy) { }
    void Set(double xx, double yy) {
        x = xx, y = yy;
    }
    // ...
};
```

Đối tượng là thành phần của mảng

```
class String{
    char *p;
public:
    String(char *s = "") { p = strdup(s); }
    String(const String &s) { p = strdup(s.p); }
    ~String() {
        cout << "delete "<< (void *)p << "\n";
        delete [] p;
    }
};
```

Thiết lập với tham số mặc nhiên

```
class SinhVien{
    String MaSo, HoTen;
    int NamSinh;
public:
    SinhVien(char *ht="Nguyen Van A", char *ms="19920014", int ns =
    1982) : HoTen(ht), MaSo(ms), NamSinh(ns) { }
};
String as[3];
Diem ad[5];
SinhVien asv[7];
```

Thiết lập không tham số

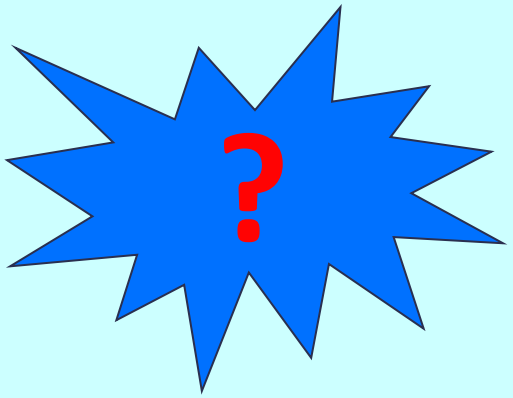
```
class Diem
{
    double x,y;
public:
    Diem(double xx, double yy) : x(xx), y(yy)
    { }
    Diem() : x(0), y(0)
    { }
    // ...
};
```

Thiết lập không tham số

```
class String{
    char *p;
public:
    String(char *s) { p = strdup(s); }
    String() { p = strdup(“”); }
    ~String() {
        cout << "delete "<< (void *)p << "\n";
        delete [] p;
    }
};
```

Thiết lập không tham số

```
class SinhVien {
    String MaSo, HoTen;
    int NamSinh;
public:
    SinhVien(char *ht, char *ms, int ns) : HoTen(ht), MaSo(ms),
    NamSinh(ns) { }
    SinhVien() : HoTen("Nguyen Van A"), MaSo("19920014"),
    NamSinh(1982) { }
};
String as[3];
Diem ad[5];
SinhVien asv[7];
```



1. Đối tượng là thành phần của lớp
2. Đối tượng là thành phần của mảng
3. **Cấp phát động đối tượng**

Đối tượng được cấp phát động

- Đối tượng được cấp phát động là các đối tượng được tạo ra bằng phép toán **new** và bị hủy đi bằng phép toán **delete**

- Phép toán new cấp đối tượng trong vùng heap và gọi phương thức thiết lập cho đối tượng được cấp

Đối tượng được cấp phát động

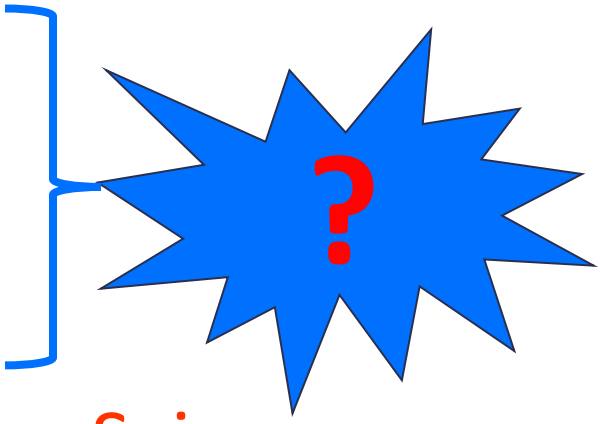
```
class String {
   char *p;
public:
   String( char *s ) { p = strdup(s); }
   String( const String &s ) { p = strdup(s.p); }
   ~String() { delete [] p; }
   //...
};
class Diem {
   double x,y;
public:
   Diem(double xx, double yy) : x(xx), y(yy) {  }
   //...
};
```

Cấp phát và hủy một đối tượng

```
int *pi = new int;
int *pj = new int(15);
Diem *pd = new Diem(20,40);
String *pa = new String("Nguyen Van A");
//...
delete pa;
delete pd;
delete pj;
delete pi;
```

Cấp phát và hủy nhiều đối tượng

- int *pai = new int[10];
- Diem *pad = new Diem[5];
- String *pas = new String[5];



Sai

- Trong trường hợp cấp phát nhiều đối tượng, ta không thể cung cấp tham số cho từng phần tử được cấp phát

Cấp và hủy nhiều đối tượng

- Thông báo lỗi cho đoạn chương trình trên như sau:
  - Cannot find default constructor to initialize array element of type 'Diem'
  - Cannot find default constructor to initialize array element of type String’
- Khắc phục lỗi?

Lỗi trên được khắc phục bằng cách cung cấp phương thức thiết lập để đối tượng có khả năng tự khởi động

Cấp và hủy nhiều đối tượng

```
class String{
    char *p;
public:
    String (char *s = "Alibaba") { p = strdup(s); }
    String (const String &s) { p = strdup(s.p); }
    ~String () {delete [] p;}
    //...
};
class Diem {
    double x,y;
public:
    Diem (double xx, double yy) : x(xx),y(yy){};
    Diem () : x(0),y(0){};
};
```

Cấp và hủy nhiều đối tượng

- Khi đó mọi phần tử được cấp đều được khởi động với cùng giá trị
  - int *pai = new int[10];
  - Diem *pad = new Diem[5]; //Ca 5 diem co cung toa do (0,0)
  - String *pas = new String[5]; //Ca 5 chuoi cung duoc khoi dong la "Alibaba"

Cấp và hủy nhiều đối tượng

- Việc hủy nhiều đối tượng được thực hiện bằng cách dùng delete và có thêm dấu [] ở trước
  - delete [] pas;
  - delete [] pad;
  - delete [] pai;

Tóm tắt về tạo đối tượng

❖Đối tượng là thành phần của lớp

- ❑Là đối tượng được tạo nên từ các đối tượng thuộc lớp khác, khi một đối tượng thuộc lớp **"lớn" (TamGiac)** được tạo ra, các thành phần (**Diem**) của nó cũng được tạo ra
- ❑Phương thức thiết lập (nếu có) sẽ được tự động gọi cho các đối tượng thành phần
  - Nếu khởi tạo lớp "lớn" không chỉ định khởi tạo nào của các đối tượng thành phần được gọi thì khởi tạo mặc định của lớp thành phần được gọi
  - Ngược lại, lớp "lớn" có thể chỉ định khởi tạo của các đối tượng thành phần và dùng dấu hai chấm (:) theo sau bởi tên thành phần và tham số khởi tạo tượng ứng
- ❑Khi đối tượng "lớn" bị hủy **(trước)** thì đối tượng thành phần của nó cũng bị hủy **(sau)**, nghĩa là phương thức hủy bỏ sẽ được gọi cho các đối tượng thành phần, sau khi phương thức hủy bỏ của đối tượng kết hợp được gọi

Tóm tắt về tạo đối tượng

❖Đối tượng là thành phần của mảng

- ❑Khi một mảng được tạo ra ➔ các phần tử của nó cũng được tạo ra ➔ phương thức thiết lập sẽ được gọi cho từng phần tử
- ❑Vì không thể cung cấp tham số khởi động cho tất cả các phần tử của mảng khi khai báo mảng, mỗi đối tượng trong mảng phải có khả năng tự khởi động, nghĩa là có thể thiết lập không cần tham số ➔ lớp phải có thể khởi tạo mặc định
- ❑Đối tượng có khả năng tự khởi động trong những trường hợp nào?
  - Lớp không có phương thức thiết lập
  - Lớp có phương thức thiết lập không tham số hoặc tất cả tham số đều là mặc nhiên

Tóm tắt về tạo đối tượng

❖Cấp phát động đối tượng

- ❑Sử dụng bộ nhớ động để tạo ra các đối tượng mà kích thước hoặc số lượng không biết trước trong thời gian biên dịch
- ❑Được thực hiện bằng cách sử dụng toán tử new để cấp phát bộ nhớ và toán tử delete để giải phóng bộ nhớ sau khi đối tượng không còn cần thiết
- ❑Sau khi đối tượng không còn cần thiết, bạn phải sử dụng toán tử delete để giải phóng bộ nhớ, nếu không sẽ gây ra rò rỉ bộ nhớ (memory leak)
- ❑Nếu cấp phát động mảng đối tượng dạng mảng bằng new[] thì hủy bằng delete[]

Bài tập

1. Xây dựng lớp hỗn số (dựa trên lớp phân số đã có trước đó) gồm một số nguyên và một phân số. Xây dựng các phương thức khởi tạo, hủy (nếu cần), nhập, xuất, getter, setter, cộng, trừ, nhân, chia giữa hai hỗn số và viết chương trình nhập vào 2 hỗn số, kiểm tra kết quả thực hiện các thao tác và xuất ra kết quả.
2. Xây dựng lớp tam giác gồm 03 điểm (dựa trên lớp điểm đã xây dựng gồm các phương thức khởi tạo, hủy (nếu cần), nhập, xuất, tính khoảng cách, getter, setter). Xây dựng các phương thức khởi tạo, hủy (nếu cần), nhập, xuất, getter, setter, tính chu vi, diện tích của tam giác và viết chương trình nhập vào 2 tam giác, xuất ra thông tin tam giác có chu vi lớn hơn, tam giác có diện tích bé hơn.