BÀI THỰC HÀNH
LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

Bài 02: Lớp và đối tượng

Trình bày: ThS. Lê Thanh Trọng

Mục tiêu

- Bài thực hành giúp người học:
  - Ôn tập về cách khai báo và sử dụng lớp và đối tượng, cùng các vấn đề quan trọng/ thường gặp trong xây dựng lớp và đối tượng
  - Hiểu rõ hơn nội dung liên quan thông qua các mã nguồn minh họa về xây dựng lớp và đối tượng
  - Vận dụng giải quyết một số bài toán về xây dựng và sử dụng lớp và đối tượng

NỘI DUNG

1. Tóm tắt kiến thức liên quan
2. Bài tập minh họa
3. Bài tập vận dụng

NỘI DUNG

1. Tóm tắt kiến thức liên quan
2. Bài tập minh họa
3. Bài tập vận dụng

Tóm tắt kiến thức liên quan

❖ Phạm vi truy cập

❑**private**: chỉ các phương thức của lớp và hàm bạn, lớp bạn được truy cập (*)

- ➢ Là phạm vi truy cập mặc định
- ➢ Thông thường thiết lập cho các thuộc tính

```
class Diem
{
private:
    float TungDo;
    float HoanhDo;
```

❑**protected**:

- ➢ (*) và lớp con/ cháu có thể truy cập
- ➢ Muốn chia sẻ thêm cho các lớp con/ cháu (kế thừa)

```
class Diem
{
protected:
    Color Clr;
```

❑**public**:

- ➢ có thể được truy cập bất kể ở đâu
- ➢ Thông thường thiết lập cho các phương thức

```
class Diem
{
public:
    Diem(float = 0, float = 0);
    void Nhap();
    void Xuat();
```

Tóm tắt kiến thức liên quan

❖Constructor

- ❑Tên phương thức **trùng với tên lớp**
- ❑Không có giá trị trả về (kể cả void)
- ❑Có thể được khai báo chồng, có thể có các tham số mặc nhiên
- ❑Thường có phạm vi là **public**

```
Diem(float = 0, float = 0);
```

```
Diem Diem1, Diem2 (0,0), Diem3 (0);
```

```
PhanSo(int _Tu = 0, int _Mau = 1);
```

```
PhanSo arrPhanSo[20];
```

```
Vector(const Vector&);
```

```
Vector* pVT = new Vector();
```

```
Vector* arrVT = new Vector[20];
```

Tóm tắt kiến thức liên quan

❖Destructor

- ❑ Tên trùng tên với tên lớp nhưng
- ❑ Có dấu ~ đặt trước, không có kiểu trả về (***~ClassName()***)
- ❑Hủy khi:

```
{
    Vector VT;
    // Sử dụng đối tượng
}
```

Hết phạm vi khai báo
(biến, mảng đối tượng)

```
Vector* pVT = new Vector();
// Sử dụng đối tượng *pVT

if(pVT!=NULL)
    delete pVT;
```

Thu hồi vùng nhớ cấp phát động
(con trỏ)

Tóm tắt kiến thức liên quan

❖ Con trỏ this

- ❑Đại diện cho đối tượng ngầm định gọi phương thức
- ❑Tham chiếu đến thành phần
  - ➢ this ➔ x
  - ➢ this ➔ GoiHam();

Tóm tắt kiến thức liên quan

❖ Thành phần static

- ❑Thuộc về lớp chứ không phải thuộc về các đối tượng cụ thể
- ❑Hàm static chỉ có thể truy cập các biến tĩnh và các hàm tĩnh khác trong lớp, không thể truy cập các biến và hàm thành viên không tĩnh.

```
class PhanSo
{
public:
    static int SoPhanSo;
    static int TimUCLN(int, int);
```

```
int main()
{
    cout << PhanSo::SoPhanSo;
    PhanSo::TimUCLN(6, 4);
```

Tóm tắt kiến thức liên quan

❖ Hàm/ lớp bạn

- ❑Là hàm/ lớp có quyền truy cập vào các thành viên riêng tư (private và protected) của một lớp khác
- ❑Dùng friend để khai báo

```
friend PhanSo TinhTong(const PhanSo&, const PhanSo&);
```

```
class PhanSo
{
public:
    friend class HonSo;
```

```
HonSo::HonSo(PhanSo& PS)
{
    PhanNguyen = PS.TuSo / PS.MauSo;
    this->PS.SetMau(PS.MauSo);
    this->PS.SetTu(PS.TuSo - PhanNguyen * PS.MauSo);
    this->PS.RutGon();
}
```

Tóm tắt kiến thức liên quan

❖Đối tượng là thành phần của lớp

- ❑Lớp **"lớn" (TamGiac)** được tạo ra, các thành phần con (**Diem**) của nó cũng được tạo ra
- ❑Phương thức thiết lập (nếu có) sẽ được tự động gọi cho các đối tượng thành phần
- ❑Khi đối tượng "lớn" bị hủy **(trước)** thì đối tượng thành phần của nó cũng bị hủy **(sau)**

```
class HonSo
{
    int PhanNguyen;
    PhanSo PS;
```

```
HonSo::HonSo(int _PhanNguyen, int _TuSo, int _MauSo):PS(_TuSo, _MauSo)
{
    PhanNguyen = _PhanNguyen;
    this->RutGon();
}
```

Tóm tắt kiến thức liên quan

❖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 ➔ lớp có khả năng khởi tạo mặc định

❖ Cấp phát động các đối tượng

- ❑Cấp phát và hủy 1 đối tượng: new, delete
- ❑Cấp phát và hủy mảng đối tượng new[], delete[]

```
Vector* pVT = new Vector();
// Sử dụng đối tượng *pVT
if(pVT!=NULL)
    delete pVT;
```

```
Vector* arrVT = new Vector[20];
// Sử dụng các đối tượng arrVT
if (arrVT != NULL)
    delete[]arrVT;
```

NỘI DUNG

1. Tóm tắt kiến thức liên quan
2. **Bài tập minh họa**
3. Bài tập vận dụng

Bài tập minh họa

- Xây dựng lớp Điểm (Point) trong hình học 2D với các thuộc tính: tung độ (float), hoành độ (float) và các thao tác (phương thức):
  - Khởi tạo với 2 tham số mặc nhiên (0 ; 0)
  - Nhập, Xuất
  - **Getter** và **Setter** cho các thuộc tính
  - Tính khoảng cách với một điểm
- Xây dựng lớp đoạn thẳng (gồm 2 điểm) và các phương thức khởi tạo (mặc định 2 điểm (0,0) và (0,0)), nhập, xuất, tính khoảng cách.
- Viết chương trình nhập vào n đoạn thẳng, xuất ra thông tin các đoạn thẳng vừa nhập và xuất ra thông tin đoạn thẳng có chiều dài nhỏ nhất và lớn nhất.

Chương trình gợi ý

```
//Diem.h
class Diem
{
private:
        float TungDo;
        float HoanhDo;
public:
        Diem(float = 0, float = 0);
        void Nhap();
        void Xuat();

        float GetTungDo();
        void SetTungDo(float);

        float GetHoanhDo();
         void SetHoanhDo(float);

        float TinhKhoangCach(const Diem&);
};
```

```
//DoanThang.h
#include"Diem.h"

class DoanThang
{
private:
        Diem A, B;
public:
        DoanThang(float XA = 0, float YA =
0, float XB = 0, float YB = 0);

        void Nhap();
        void Xuat();

        float TinhKhoangCach();
};
```

Chương trình gợi ý

```
//main.cpp
#include<iostream>
#include"DoanThang.h"

using namespace std;

int main()
{
        DoanThang* arrDoanThang;
        int n;
        cout << "Nhap so doan thang = ";
        cin >> n;
        arrDoanThang = new DoanThang[n];
        for (int i = 0; i < n; i++)
        {
                cout << "Nhap doan thang " << i + 1;
                arrDoanThang[i].Nhap();
        }
        cout << "Cac doan thang vua nhap la: " << endl;
        for (int i = 0; i < n; i++)
        {
                cout << endl << "Doan thang " << i + 1 <<": ";
                arrDoanThang[i].Xuat();
                cout << endl;
        }
```

Chương trình gợi ý

```
float minKhoangCach = arrDoanThang[0].TinhKhoangCach();
float maxKhoangCach = arrDoanThang[0].TinhKhoangCach();

int minIndex = 0, maxIndex = 0;
for (int i = 1; i < n; i++)
{
        if (minKhoangCach > arrDoanThang[i].TinhKhoangCach())
        {
                minKhoangCach = arrDoanThang[i].TinhKhoangCach();
                minIndex = i;
        }
        if (maxKhoangCach < arrDoanThang[i].TinhKhoangCach())
        {
                maxKhoangCach = arrDoanThang[i].TinhKhoangCach();
                maxIndex = i;
        }
}
cout << endl << "Doan thang ngan nhat la: ";
arrDoanThang[minIndex].Xuat();

cout << endl << "Doan thang dai nhat la: ";
arrDoanThang[maxIndex].Xuat();

return 1;
}
```

NỘI DUNG

1. Tóm tắt kiến thức liên quan
2. Bài tập minh họa
3. **Bài tập vận dụng**

Bài tập vận dụng

1. Viết định nghĩa lớp biểu diễn khái niệm thời gian với các thành phần dữ liệu giờ, phút, giây với các thao tác thích hợp (khởi tạo, nhập, xuất, tính tổng và so sánh hai thời gian). Viết chương trình nhập vào n (do người dùng nhập vào) đối tượng thời gian và xuất danh sách thời gian theo thứ tự tăng dần, tổng các đối tượng thời gian vừa nhập.
2. Xây dựng lớp Candidate (Thí sinh) gồm các thuộc tính: mã, tên, ngày tháng năm sinh, điểm thi Toán, Văn, Anh và các phương thức cần thiết (khởi tạo, nhập, xuất, tính tổng điểm). Xây dựng lớp TestCandidate quản lý n thí sinh và các phương thức thích hợp (khởi tạo, nhập, xuất, sắp xếp điểm tổng tăng dần). Viết chương trình nhập vào n thí sinh, in ra thông tin về các thí sinh có điểm tổng tăng dần.

Bài tập vận dụng

3. Định nghĩa lớp **DSPhanSo** (danh sách phân số) để lưu trữ và xử lý các thao tác trên mảng các phân số. Viết chương trình cho phép người dùng nhập vào danh sách các phân số và thực hiện các thao tác

- Tính và xuất tổng các phân số
- Xuất phân số lớn nhất
- Xuất phân số nhỏ nhất
- Sắp xếp và xuất danh sách phân số tăng dần
- Sắp xếp và xuất danh sách phân số giảm dần

Bài tập vận dụng

4. Xét đa thức theo biến x (đa thức một biến) bậc n có dạng như sau:

$P(X) = a_1x^n + a_2x^{n-1} + a_3x^{n-2} + ... + a_j$

Trong đó: n là bậc của đa thức. a_1, $a_{2,}$ a_3,..., a_j là các hệ số tương ứng với từng

bậc của đa thức.

Định nghĩa lớp DaThuc biểu diễn khái niệm đa thức với các thao tác sau:

- Khởi tạo một đa thức có bậc bằng 0 hoặc bậc.
- Tính giá trị của đa thức khi biết giá trị của x
- Nhập đa thức
- Xuất đa thức
- Cộng hai đa thức
- Trừ hai đa thức

Viết chương trình cho phép người dùng nhập vào hai đa thức, xuất các đa thức ra màn hình, tính tổng, hiệu hai đa thức và xuất kết quả ra màn hình.

CHÚC CÁC BẠN LÀM BÀI TỐT