

Lab 02

Hành vi của lớp

Lập trình hướng đối tượng

Mục tiêu

1. Cài đặt hành vi của đối tượng
2. Cài đặt getter phái sinh
3. Cấp phát và thu hồi vùng nhớ cho thành phần

1 Hướng dẫn khởi đầu 1 - Hành vi

Mô tả bài tập

Cho trước thiết kế lớp **Điểm** trong không gian hai chiều với 2 thuộc tính **x** và **y**.

Cài đặt hàm tính khoảng cách từ một điểm đến một điểm khác

Point	CPoint
-_x: float -_y: float	-m_fx: float -m_fY: float
+X(): float +Y(): float +SetX(float) +SetY(float) +Point() +Point(float, float) ~Point() +CalcDistanceTo(const Point*)	+GetX(): float +GetY(): float +SetX(float) +SetY(float) +CPoint() +CPoint(float, float) ~CPoint() +CalcDistanceTo(const Point*)

Thiết kế tinh gọn và theo kí pháp Hungary

- Tạo ra định nghĩa lớp như sau:

```
#pragma once
#include <math.h>

class Point {
private:
    float _x;
    float _y;
public:
    float X() { return _x; }
    float Y() { return _y; }
    void SetX(float value) { _x = value; }
    void SetY(float value) { _y = value; }
public:
    Point();
    Point(float, float);
    ~Point();
public:
    float CalcDistanceTo(const Point* other);
};
```

Chú ý từ khóa `const` ám chỉ ta không muốn thay đổi giá trị của thành phần kiểu dữ liệu `Point` mà con trỏ `other` đang trỏ đến.

Bước 3: Cài đặt lớp trong file Point.cpp (CPoint.h)

```
#include "Point.h"

]Point::Point() {
    this->_x = 0;
    this->_y = 0;
}

]Point::Point(float x, float y) {
    this->_x = x;
    this->_y = y;
}

]Point::~~Point() {
}

]float Point::CalcDistanceTo(const Point* other) {
    float dx = this->_x - other->_x;
    float dy = this->_y - other->_y;

    return sqrt(dx * dx + dy * dy);
}
```

Chú ý bên trong hàm CalcDistanceTo có thể truy xuất thành phần private của đối tượng con trả other đang trả về. Vì cả hai (this và other) đều trả về **cùng một loại** đối tượng nên **hàm bên trong lớp Point** có thể truy xuất đến thuộc tính private, vốn đáng lẽ bị che dấu với các đối tượng khác kiểu.

Bước 4: Cài đặt hàm main để test việc cài đặt của lớp Point (CPoint)

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

#include "Point.h"

int main()
{
    Point* start = new Point(4, 3);
    Point* end = new Point(10, 9);
    float length = start->CalcDistanceTo(end);
    cout << "Khoang cach hai diem la: " << length << endl;
    delete start;
    delete end;
}
```

Chạy lên và thấy kết quả như sau:

```
Khoang cach hai diem la: 8.48528
```

② Hướng dẫn khởi đầu 2 - Getter phái sinh

Mô tả bài tập

Cho trước thiết kế lớp **Đường tròn** trong không gian hai chiều với 2 thuộc tính **Tâm** và **Bán kính**.

Cài đặt hàm lấy các thông tin đặc trưng của đường tròn:

- + Đường kính (Diameter)
- + Chu vi (Perimeter)
- + Diện tích (Area)

Hướng dẫn cài đặt

Bước 1: Thêm mới dự án

- Thêm mới một dự án vào solution và đặt tên là **CircleV2**.

Bước 2: Tạo định nghĩa lớp trong file **Circle.h**

- Thêm một tập tin header bằng cách nhấn phải vào project, chọn **Add > New Item...**
- Chọn loại tập tin là Header file và đặt tên là **Circle.h**
- Nội dung khai báo lớp như bên dưới:

```
class Circle {  
public:  
    const int PI = 3.14;  
private:  
    Point* _center;  
    float _radius;  
public:  
    Point* Center() { return _center; }  
    float Radius() { return _radius; }  
    void SetCenter(Point* value) { _center = value; }  
    void SetRadius(float value) { _radius = value; }  
public:  
    float Diameter() { return 2 * _radius; }  
    float Perimeter() { return 2 * _radius * PI; }  
    float Area() { return _radius * _radius * PI; }  
};
```

Chú ý: Tên hàm thường bắt đầu bằng động từ. Khi ta gặp tên hàm bắt đầu bằng danh từ thì có thể hiểu đó là hàm lấy property.

3 Hướng dẫn khởi đầu 3 - Cấp phát và thu hồi

Mô tả bài tập

Cải tiến cài đặt lớp **Đường tròn**, trong hàm tạo cấp phát vùng nhớ cho lớp Điểm, trong hàm hủy thu hồi vùng nhớ đã cấp phát

Hướng dẫn cài đặt

Bổ sung thêm hai cài đặt như sau:

```
class Circle {
public:
    const int PI = 3.14;
private:
    Point* _center;
    float _radius;
public:
    Point* Center() { return _center; }
    float Radius() { return _radius; }
    void SetCenter(Point* value) { _center = value; }
    void SetRadius(float value) { _radius = value; }
public:
    float Diameter() { return 2 * _radius; }
    float Perimeter() { return 2 * _radius * PI; }
    float Area() { return _radius * _radius * PI; }
public:
    Circle() {
        _center = new Point();
    }

    ~Circle() {
        delete _center;
    }
};
```

Chú ý: Cài đặt đúng cần tách ra file .h và .cpp riêng! Tự viết hàm main kiểm tra lớp!

4 Bài tập vận dụng

Yêu cầu

1. Thực hiện định nghĩa lớp theo thiết kế cho trước vào tập tin .h.
2. Thực hiện cài đặt lớp trong tập tin .cpp cho lớp tương ứng.
 1. Thuộc tính có đầy đủ getter / setter
 2. Có hàm tạo, hàm hủy thực hiện cấp phát và thu hồi các thành phần
3. Viết các đoạn mã nguồn kiểm tra việc định nghĩa lớp trong hàm main.

Danh sách bài tập cụ thể

1. Lớp **Hình chữ nhật** có hai thành phần **Điểm**: **Trái trên** và **Phải Dưới**

Gợi ý: Tên project: RectangleV2

- + Tên lớp: Rectangle / CRectangle
- + Thành phần: _topLeft, _bottomRight
- + Chu vi, diện tích
- + Hàm tạo mặc định khởi tạo 2 điểm ở tọa độ (1, 1) và (10, 10).
- + Hàm hủy: thu hồi vùng nhớ của _topLeft và _bottomRight

2. Lớp **Hình tam giác** có ba thành phần **Điểm** ứng với 3 đỉnh : **a, b, c**.

Gợi ý: Tên project: TriangleV2

- + Tên lớp: Triangle / CTriangle
- + Thành phần: _a, _b, _c
- + Chu vi, diện tích
- + Hàm tạo mặc định khởi tạo 3 điểm (1, 1), (1, 2), (4, 2)
- + Hàm hủy thu hồi vùng nhớ 3 điểm thành phần a, b, c.

3. Lớp **Sinh viên** có 3 thành phần: **họ** (chuỗi), **tên lót** (chuỗi) và **tên** (chuỗi).

Gợi ý: Tên project StudentV2

- + Tên lớp: Student / CStudent
- + Thành phần: _firstName, _middleName, _lastName. Kiểu dữ liệu: string. Cần include <string>
- + Họ tên (đầy đủ): gợi ý tên Fullname. Tạo ra từ 3 thành phần first, middle, last. Chỉ cần sử dụng toán tử + của lớp string là đủ
- Return _firstName + " " + _middleName + " " + _lastName
- + Hàm hủy: không cần làm gì

4. Lớp **Mảng động** (**DynamicArray**) có 3 thành phần

- + `int* _a`: chứa mảng các số nguyên
- + `int _len`: chứa độ dài hiện tại của mảng
- + `int _max`: chứa độ dài tối đa mảng hỗ trợ

Khi khởi tạo, tạo mảng mặc định có 128 phần tử.

Khi hủy, nhớ thu hồi vùng nhớ cấp phát cho mảng

Thành phần cơ bản gồm:

+ `PushBack(int value)` : thêm 1 phần tử vào mảng. Cài đặt hiện tại không cần tính tới việc vượt quá khả năng của mảng. Ta sẽ cài ở tuần sau. Cứ việc đẩy phần tử mới vào mảng ở phiên bản này.

+ `GetAt(int i)`: Lấy một phần tử tại vị trí `i`. Cài đặt hiện tại không cần tính tới việc truy cập phần tử không hợp lệ. Ta sẽ cài đặt ở tuần sau. Cứ việc trả ra phần tử tại vị trí thứ `i`.

```
class DynamicArray {  
private:  
    const int INITIAL_SIZE = 128;  
private:  
    int* _a;  
    int _len;  
    int _max;  
public:  
    void PushBack(int value);  
    int GetAt(int index);  
};
```

5 Hướng dẫn nộp bài

Tổ chức bài nộp

- + Mỗi bài tương ứng với 1 folder. Chỉ chứa file .h và .cpp, và Makefile
- + Tổng hợp tất cả folder vào folder MSSV.
- + Nén lại tất cả thành một tập tin duy nhất MSSV.zip.

Để nộp bài, nén tất cả lại và đặt tên với định dạng **MSSV.zip**

-- HẾT --