Lab 03
Thành phần tĩnh

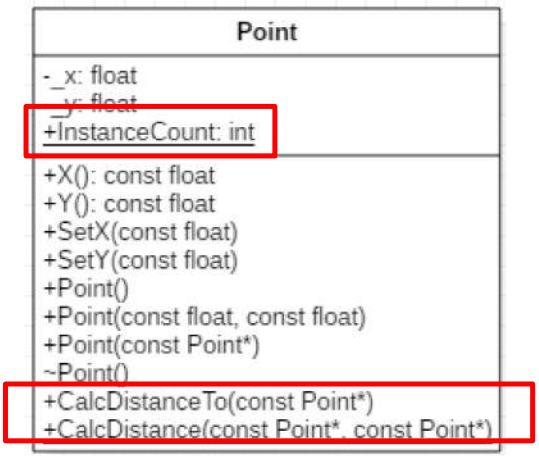
Lập trình hướng đối tượng

Mục tiêu	1. Cài đặt và sử dụng thuộc tính tĩnh 2. Cài đặt và sử dụng hàm tĩnh

1 Hướng dẫn khởi đầu

Mô tả bài tập

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



Hãy cài đặt cụ thể lớp này với các thành phần:

+ Thuộc tính private

+ Các hàm getter setter tương ứng. *Chú ý từ khóa **const** báo hiệu chỉ muốn truy cập giá trị mà không muốn thay đổi*

+ Hàm tạo và hàm hủy

+ Hàm tạo có đối số

+ Hàm CalcDistanceTo để tính khoảng cách đến điểm khác

*+ Thành phần **tĩnh InstanceCount** đếm số lượng thể hiện đã tạo ra của lớp Điểm*

*+ Hàm **tĩnh CalcDistance** để tính khoảng cách giữa hai điểm*

- Tạo ra định nghĩa lớp như sau trong file Point.h

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

class Point {
public:
    static int InstanceCount;
private:
    float _x;
    float _y;
public:
    const float X() { return _x; }
    const float Y() { return _y; }
    void SetX(const float value) { _x = value; }
    void SetY(const float value) { _y = value; }
public:
    Point();
    Point(const float, const float);
    ~Point();
public:
    float CalcDistanceTo(const Point* other) const;
public:
    static float CalcDistance(const Point* a, const Point* b);
};

int Point::InstanceCount = 0;
```

Chỉ nên đặt trước hàm main!

Đặt ở đây sẽ báo lỗi

Việc sử dụng const đối với các kiểu dữ liệu nguyên thủy như int, float,v.v là không cần thiết vì bản chất đã là truyền giá trị và không thể thay đổi đối số. Hệ quả là lớp điểm các hàm getter / setter không nhất thiết phải có const. Tuy nhiên nếu các thành phần là con trỏ thì cần chú ý điểm này.

Bước 3: Cài đặt lớp trong file Point.cpp

```cpp
#include "Point.h"

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

    Point::InstanceCount++;
}

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

    Point::InstanceCount++;
}

Point::~Point() { }

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

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

float Point::CalcDistance(const Point* a, const Point* b) {
    return a->CalcDistanceTo(b);
}
```

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

```
int Point::InstanceCount = 0;
int main()
{
    Point* start = new Point(4, 3);
    Point* end = new Point(10, 9);
    float length = Point::CalcDistance(start, end);
    cout << "Khoang cach hai diem la: " << length << endl;

    cout << "So diem da tao ra:" << Point::InstanceCount << endl;
    delete start;
    delete end;
}
```

Chạy lên và thấy kết quả như sau:

```
Khoang cach hai diem la: 8.48528
So diem da tao ra:2
```

2 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.
3. Viết các đoạn mã nguồn kiểm tra việc định nghĩa lớp trong hàm main.

Danh sách bài tập cụ thể

1. Lớp **Đường thẳng** có hai thành phần **Điểm**: **Bắt đầu** và **Kết thúc**.

Gợi ý: Tên project: **LineV3**

+ Tên lớp: Line / CLine

+ Thành phần: _start, _end

+ Thuộc tính: Length cho biết độ dài của đường thẳng

+ Thuộc tính **tĩnh** *InstanceCount* cho biết đã tạo ra bao nhiêu đối tượng từ lớp đường thẳng.

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

Gợi ý: Tên project: **RectangleV3**

+ Tên lớp: Rectangle / CRectangle

+ Thành phần: _topLeft, _bottomRight

+ Thuộc tính **tĩnh** *InstanceCount* cho biết đã tạo ra bao nhiêu đối tượng từ lớp hình chữ nhật

3. Lớp **Random** sinh ra một số nguyên ngẫu nhiên.

Gợi ý: Tên project **RandomV3**

+ Tên lớp: Random

+ Thành phần: không có

+ Hàm tạo: Khởi tạo bộ sinh số ngẫu nhiên

+ Hàm **Next**(): trả ra một số nguyên

+ Hàm **Next**(int max): trả ra một số nguyên từ 0 đến max-1 (Gợi ý: dùng toán tử %)

4. Lớp Xúc sắc (**Dice**). Gợi ý tên project **DiceV3**

+ Thuộc tính: Không có

+ Hàm **Roll**(): Thực hiện gieo xúc sắc và trả ra 1 trong 6 giá trị từ 1 đến

6 cho biết đã gieo được mặt nào của xúc sắc

+ Thuộc tính _rollCount cho biết đã gieo xúc sắc bao nhiêu lần. Chú ý thuộc tính này không phải thuộc tính tĩnh

Gợi ý: Tạo ra 3 xúc sắc, mỗi xúc sắc gieo một số lần ngẫu nhiêu, sau đó thống kê mỗi loại xúc sắc đã gieo bao nhiêu lần.

3 Hướng dẫn nộp bài

Tổ chức bài nộp

+ Mỗi bài tương ứng với 1 folder. Chỉ chứa file .h và .cpp, Makefile và file .jpg chứa sơ đồ UML.

+ 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 --