Lab 03

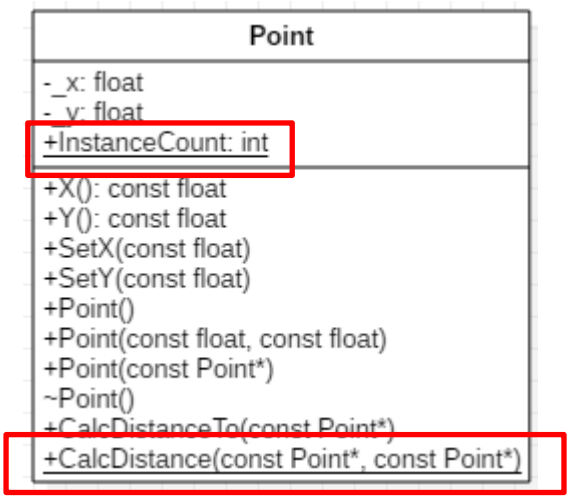
Thành phần tĩnh & Hàm tạo sao chép

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*

Hướng dẫn cài đặt

Bước 1: Tạo định nghĩa lớp trong file Point.h

- Thêm một tập tin header bằng cách nhấn phải vào project, chọn **Add > New Item**...

- Chọn loại tập tin là **Header File (.h),** đặt tên là Point.h

- Tạo ra định nghĩa lớp như sau:

```
#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 2: 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 3: 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
```

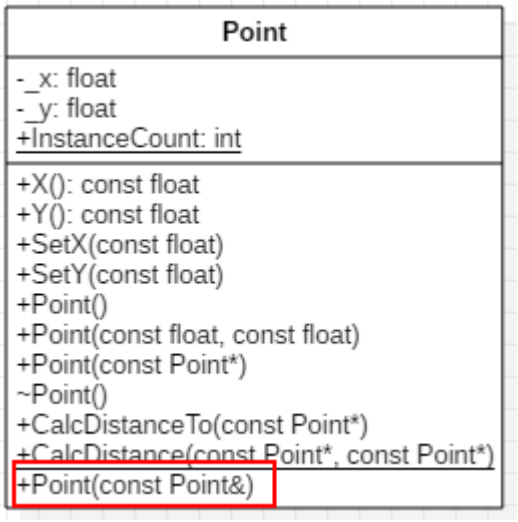
2 Hàm tạo sao chép

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.
+ 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

+ Hàm tạo sao chép để khởi tạo thông tin từ một điểm khác

Bước 1: Tạo định nghĩa lớp trong file Point.h (CPoint.h)

- Thêm một tập tin header bằng cách nhấn phải vào project, chọn **Add > New Item**...

- Chọn loại tập tin là **Header File (.h),** đặt tên là Point.h

- Tạo ra định nghĩa lớp như sau:

```
#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();
    Point(const Point&);
public:
    float CalcDistanceTo(const Point* other) const;
public:
    static float CalcDistance(const Point* a, const Point* b);
};
```

Bước 2: Cài đặt thêm hàm tạo sao chép trong file Point.cpp

```
Point::Point(const Point& other)
{
    _x = other._x;
    _y = other._y;
}
```

Bước 3: Cài đặt hàm main để test việc cài đặt của lớp Point (CPoint)

```
int Point::InstanceCount = 0;

int main()
{
    Point root(5, 6);
    Point copy(root);
    cout << "Nut copy:" << copy.X() << " " << copy.Y() << " " << endl;

    Point* start = new Point(4, 3);
    Point* end = new Point(10, 9);

    Point* temp = start; // HÀM TẠO SAO CHÉP KHÔNG ĐƯỢC GỌI

    // Toán tử * là dereference, biến start kiểu Point * thành Point
    Point* meet = new Point(*start); // Gọi hàm tạo sao chép
    cout << "Dia diem gap: " << meet->X() << " " << meet->Y() << " " << endl;

    delete meet;
    delete start;
    delete end;
}
```

Chạy lên và thấy kết quả như sau:

```
Nut copy:5 6
Dia diem gap: 4 3
```

3 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**.

+ 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.

+ Hàm tạo sao chép để tạo ra đường thẳng từ một đường thẳng khác

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

+ 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

+ Hàm tạo sao chép để tạo ra hình chữ nhật từ một hình chữ nhật khác

3. 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 mặc định, tạo mảng null. 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, mảng sẽ tự cơi nới thêm một phần tử.

+ 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.

+ Hàm tạo sao chép để tạo ra một mảng động khác từ một mảng động trước đó.