Môn PPLTHĐT

Hướng dẫn thực hành tuần 2

Mục đích

Giới thiệu sơ lược về đối tượng và lớp đối tượng, cách cấp phát và giải phóng vùng nhớ trong C++, tìm hiểu một số khái niệm liên quan đến đối tượng: hàm dựng, hàm hủy.

Nội dung

- Đối tượng và lớp đối tượng.
- Toán tử new và delete.
- Hàm dựng và hàm hủy.

Yêu cầu

Nắm vững những nội dung trình bày trong bài hướng dẫn thực hành số 1.

1. Đối tượng và lớp đối tượng

Khái niệm

Đối tượng *(object)* là một thực thể phần mềm bao gồm **dữ liệu** và những **xử lý trên dữ liệu** đó. Thành phần dữ liệu được gọi là thuộc tính *(attribute)*, thành phần xử lý được gọi là phương thức *(method)*.

Thuộc tính ẩn chứa bên trong đối tượng mà những gì bên ngoài không thể truy xuất đến được. Phương thức là cách thức duy nhất để những gì bên ngoài đối tượng thực hiện những xử lý trên thuộc tính và thông qua những xử lý này thuộc tính của đối tượng thay đổi.

Tập hợp các đối tượng có cùng kiểu thuộc tính và phương thức tạo thành lớp đối tượng (class). Trong C++, lớp đối tượng được khai báo theo cấu trúc sau:

class <*Tên lớp*>
{
private:
    <*Khai báo thành phần riêng*>
public:
    <*Khai báo thành phần công cộng*>
};

Ví dụ xây dựng lớp đối tượng đơn giản:

```
class SampleClass
{
private:
    // Thuoc tinh cua lop SampleClass.
    int     m_iVar1;
    float   m_fVar2;

public:
    // Phuong thuc cua lop SampleClass.
    void f1()
    {
        // Viet xu ly cho f1 o day.
    }

    void f2()
    {
        // Viet xu ly cho f2 o day.
    }
};
```

Tầm vực

Tầm vực (scope) của các thành phần bên trong lớp đối tượng định nghĩa phạm vi hoạt động của những thành phần đó. Có 3 loại tầm vực dành cho các thành phần bên trong lớp đối tượng: public, private, và protected. Ở đây chúng ta tạm thời chỉ đề cập đến 2 tầm vực public và private.

Ý nghĩa của các tầm vực này như sau:

	Sử dụng bên trong lớp	Sử dụng bên ngoài lớp
Public	X	X
Private	X	

```
class SampleClass
{
private:
	int	m_iPrivateVar;

	void f1()
	{
		// Viet xu ly cho f1 o day.
	}

public:
	int	m_iPublicVar;

	void f2()
	{
		m_iPrivateVar = 1;	// Thuoc tinh va phuong thuc private
		f1();			// co the truy cap trong pham vi lop.
	}
};

void main()
{
	SampleClass	obj;

	obj.m_iPrivateVar = 1;		// Thuoc tinh va phuong thuc private
	obj.f1();			// khong the truy cap ngoai pham vi lop.

	obj.m_iPublic = 1;		// Thuoc tinh va phuong thuc public
	obj.f2();			// co the truy cap ngoai pham vi lop.
}
```

Toán tử ::

Toán tử :: hay còn gọi là toán tử phân giải miền (scope resolution operator) được sử dụng để chỉ ra một cách tường minh một thuộc tính hoặc phương thức là thuộc một lớp đối tượng nào đó.

Ví dụ:

```
Employee::m_sName        // De cap den thuoc tinh m_sName cua lop Employee.
Circle::Draw()           // De cap den phuong thuc Draw() cua lop Circle.
```

Khi định nghĩa một phương thức bên ngoài lớp, chúng ta phải sử dụng toán tử phân giải miền để chỉ ra tường minh phương thức đó thuộc lớp nào.

Ví dụ hai cách định nghĩa phương thức sau là tương đương nhau:

```
class A
{
public:
        int f();
};

int A::f()
{
        // Cai dat phuong thuc f().
}
```

```
class A
{
public:
        int f()
        {
                // Cai dat phuong thuc f().
        }
};
```

Một lớp đối tượng trong C++ thường được lưu trữ trong 2 file: *.h và *.cpp. File *.h chứa phần khai báo (decleration) của lớp đối tượng, trong khi file *.cpp chứa phần định nghĩa.

Con trỏ this

Con trỏ this được sử dụng trong phạm vi của lớp đối tượng. Nó đại diện cho con trỏ đang trỏ đến đối tượng gọi thực hiện phương thức đó.

Ví dụ:

```
#include "iostream.h"

class A
{
public:
        void ShowThis()
        {
                cout << this << endl;
        }
};

void main()
```

```
{
        A       obj1, obj2;

        obj1.ShowThis();
        obj2.ShowThis();
}
```

Áp dụng

Để hiểu rõ hơn về đối tượng và lớp đối tượng, chúng ta xét ví dụ xây dựng lớp PhanSo.

Bước 1: vào VS, tạo project dạng Console Application (Visual C++).

Bước 2: thêm vào project file PhanSo.h và khai báo lớp PhanSo.

```
class PhanSo
{
private:
        int     m_iTuSo;
        int     m_iMauSo;

public:
        int LayTuSo();
        int LayMauSo();
        void DatTuSo(int iTuso);
        void DatMauSo(int iMauSo);
        PhanSo Cong(const PhanSo &a);
};
```

Bước 3: thêm vào project file PhanSo.cpp và định nghĩa các phương thức cho lớp PhanSo.

```
#include "iostream.h"
#include "PhanSo.h"

int PhanSo::LayTuSo()
{
        return m_iTuSo;
}

int PhanSo::layMauSo()
{
        return m_iMauSo;
}

void PhanSo::DatTuSo(int iTuSo)
{
```

```
	m_iTuSo = iTuSo;
}

void PhanSo::DatMauSo(int iMauSo)
{
	m_iMauSo = iMauSo;
}

PhanSo PhanSo::Cong(const PhanSo &a)
{
	PhanSo		c;

	c.m_iTuSo = this->m_iTuSo * a.m_iMauSo + a.m_iTuSo * this->m_iMauSo;
	c.m_iMauSo = this->m_iMauSo * a.m_iMauSo;

	return c;
}
```

Bước 4: thêm vào project file main.cpp và viết đoạn chương trình sử dụng lớp PhanSo vừa tạo.

```
#include "iostream.h"
#include "PhanSo.h"

void main()
{
	PhanSo		a, b, c;

	a.DatTuSo(1);
	a.DatMauSo(2);

	b.DatTuSo(1);
	b.DatMauSo(3);

	c = a.Cong(b);

	cout << "Tu so c = " << c.LayTuSo() << endl;
	cout << "Mau so c = " << c.LayMauSo() << endl;
}
```

Bước 5: biên dịch và chạy thử chương trình.

Trong lớp PhanSo ở trên, các thuộc tính m_iTuSo và m_iMauSo được khai báo trong phần "private" nên có tầm vực là private, vì vậy chúng ta không thể truy cập được chúng từ bên ngoài lớp. Trong khi đó, những phương thức được khai báo trong phần "public" nên có tầm vực là public, do đó chúng có thể được gọi từ bên ngoài lớp.

2. Toán tử new và delete

Trước đây trong C, việc cấp phát và giải phóng vùng nhớ được thực hiện thông qua các hàm trong thư viện "alloc.h".

Ví dụ cấp phát vùng nhớ 10 byte dùng để chứa chuỗi ký tự:

```
unsigned char  *s = (unsigned char *)malloc(10);

// ...
// ...
// ...

free(s);
```

Trong C++, việc cấp phát và giải phóng vùng nhớ được tích hợp trực tiếp vào bên trong ngôn ngữ thông qua 2 toán tử new và delete.

new

Toán tử new dùng để cấp phát vùng nhớ cho một con trỏ thuộc một kiểu dữ liệu nào đó.

Ví dụ:

```
int            *p = new int[10];      // Cap phat vung nho dung de chua 10 so nguyen
                                      // cho con tro p.
unsigned char  *s = new char[10];     // Cap phat vung nho dung de chua 10 ky tu
                                      // cho con tro s.
PhanSo         *a = new PhanSo;       // Cap phat vung nho dung de chua
                                      // doi tuong PhanSo cho con tro a.
```

delete

Toán tử delete dùng để giải phóng vùng nhớ đã **được cấp phát bằng toán tử new** cho một con trỏ nào đó.

Ví dụ:

```
delete  []p;     // Giai phong vung nho da cap phat cho con tro p.
delete  []s;     // Giai phong vung nho da cap phat cho con tro s.
delete  a;       // Giai phong vung nho da cap phat cho con tro a.
```

3. Hàm dựng và hàm hủy

Hàm dựng

Hàm dựng *(constructor)* của một lớp đối tượng là một phương thức đặc biệt **được tự động gọi thực hiện** khi đối tượng thuộc lớp đó được tạo lập.

Các tính chất của hàm dựng:

- Được tự động gọi thực hiện khi đối tượng được tạo lập.
- Không có giá trị trả về (nhưng có thể có các tham số).
- Một lớp đối tượng có thể có nhiều hàm dựng khác nhau.
- Trong C++, hàm dựng có tên trùng với tên lớp.

Bất kỳ lớp đối tượng nào cũng có hàm dựng. Trong trường hợp chúng ta không khai báo hàm dựng nào cho lớp đối tượng, hàm dựng mặc định *(default constructor)* không tham số sẽ được tự động thêm vào.

Ví dụ sau thể hiện các tính chất của hàm dựng:

```
#include "iostream.h"

class SampleClass
{
public:
        SampleClass(int x)
        {
                cout << "Khoi tao doi tuong voi tham so la so nguyen.";
        }

        SampleClass(int x, char *s)
        {
                cout << "Khoi tao doi tuong voi tham so la so nguyen va chuoi ky tu.";
        }
};

void main()
{
        SampleClass   obj1(5); SampleClass
        obj2(5, "hello"); SampleClass   *p = new
        SampleClass(2);
}
```

Hàm hủy

Hàm hủy *(destructor)* của một lớp đối tượng là một phương thức đặc biệt **được tự động gọi thực hiện** khi đối tượng thuộc lớp đó bị hủy đi.

Các tính chất của hàm hủy:

- Được tự động gọi thực hiện khi đối tượng bị hủy đi.
- Không có giá trị trả về lẫn tham số.
- Một lớp đối tượng chỉ có duy nhất một hàm hủy.
- Trong C++, hàm hủy có tên trùng với tên lớp và thêm dấu "~" phía trước.

Ví dụ sau thể hiện các tính chất của hàm hủy:

```
#include "iostream.h"

class SampleClass
{
public:
	virtual ~SampleClass()
	{
		cout << "Huy bo doi tuong.";
	}
};

void main()
{
	SampleClass	obj;
	SampleClass	*p = new SampleClass;

	delete	p;
};
```