Nội dung tuần 09

Áp dụng Mẫu thiết kế hướng đối tượng.

Bài tập

❖ Yêu cầu

A: bài 1, bài 2, bài 3

H: Làm hết 5 bài

Bài 1

Cài đặt lại ví dụ 1.

Bài 2

Cài đặt lại ví dụ 2.

Bài 3

Cài đặt lại ví dụ 3.

Bài 4

Một cỗ máy được cấu tạo từ các chi tiết máy. Mỗi chi tiết máy đều được đánh mã số để phân biệt với nhau (ví dụ CT001).

Các chi tiết máy được phân làm 2 loại:

- Chi tiết đơn: không thể phân chia, có trọng lượng và giá thành.
- Chi tiết phức: cấu tạo từ các chi tiết con (đơn hoặc phức).
    - Trọng lượng = trọng lượng các chi tiết con + 10% phụ kiện kết nối.
    - Giá thành = giá thành các chi tiết con + 20% công lắp ráp.

Xây dựng các lớp đối tượng với thành phần dữ liệu và thành phần xử lý phù hợp để hàm main sau chạy đúng:

```
int main()
{
	/// tạo mới Cỗ máy
	CoMay* pcm = new CoMay();
	/// thêm chi tiết
	pcm->ThemChiTietMay(new ChiTietDon("CTD001", 120, 350000));

	/// tạo Chi tiết phức
	ChiTietMay* pctm = new ChiTietPhuc("CTP001");
	((ChiTietPhuc*)pctm)->ThemChiTiet(new ChiTietDon("CTD002", 50, 1280000));
	((ChiTietPhuc*)pctm)->ThemChiTiet(new ChiTietDon("CTD003", 20, 10000));
```

```
        ChiTietMay* pctm1 = new ChiTietPhuc("CTP002");
        ((ChiTietPhuc*)pctm1)->ThemChiTiet(new ChiTietDon("CTD004", 80, 170000));

        ((ChiTietPhuc*)pctm)->ThemChiTiet(pctm1);

        pcm->ThemChiTietMay(pctm);
        pcm->ThemChiTietMay(new ChiTietDon("CTD005", 210, 2350000));
        pcm->ThemChiTietMay(new ChiTietDon("CTD006", 40, 50000));

        /// xuất thông tin Cỗ máy
        /// xuất danh sách các chi tiết con
        cout << "Thong tin co may: " << endl << *pcm << endl;

        /// xuất trọng lượng cỗ máy
        cout << "Trong luong: " << pcm->TrongLuong() << endl;

        /// xuất giá thành cỗ máy
        cout << "Gia thanh: " << pcm->GiaThanh() << endl;

        cout << endl << endl;
        system("pause");
        return 0;
}
```

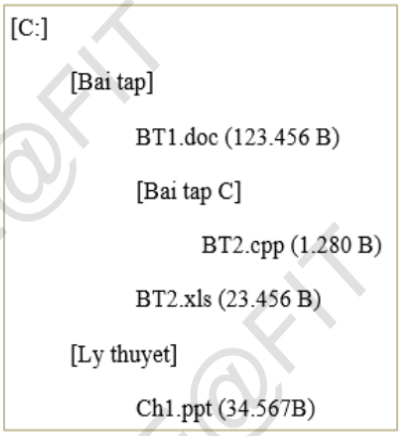
Bài 5

Đơn vị lưu trữ dữ liệu trên máy tính là tập tin và thư mục.

- Tập tin có thông tin về tên và kích thước.
- Thư mục chỉ có tên nhưng có thể chứa tập tin và thư mục khác (thư mục con).

Viết chương trình cho phép:

- Tạo một cây thư mục tập tin như bên dưới.
- Xuất cây thư mục từ thư mục gốc.
- Đếm số tập tin và thư mục có trong một thư mục nào đó.
- Tính kích thước một thư mục có tên cho trước.



Hướng dẫn

Ví dụ 1: Operator << với Polymorphism

Xét ví dụ sau với lớp đối tượng cơ sở là **Base**, lớp đối tượng dẫn xuất là **Derived1** và **Derived2** có cách xuất khác nhau. Chú ý phần highlight về cách cài đặt hàm ảo thuần túy để **Base** là abstract class tức là không được phép tạo ***instance***

```
class Base
{
public:
	virtual void Print(ostream& os) = 0;
	friend ostream& operator<<(ostream& os, Base &base)
	{
		base.Print(os);
		return os;
	}
};

class Derived1 : public Base
{
private:
	int _ofD1;
public:
	Derived1(const int& v)
	{
		_ofD1 = v;
	}

	void Print(ostream& os)
	{
		os << "D1 - " << _ofD1;
	}
};

class Derived2 : public Base
{
private:
	int _ofD2;
public:
	Derived2(const int& v)
	{
		_ofD2 = v;
	}

	void Print(ostream& os)
	{
		os << "D 2- " << _ofD2;
	}
};
```

Run với hàm main sau

```
int main()
{
	vector<Base*> ds;
```

```
	ds.push_back(new Derived1(13));
	ds.push_back(new Derived1(17));
	ds.push_back(new Derived2(24));
	ds.push_back(new Derived2(28));
	for (int i = 0; i < ds.size(); i++)
	{
		cout << *ds[i] << endl;
	}

	cout << endl << endl;
	system("pause");
	return 0;
}
```

```
D1 - 13
D1 - 17
D 2- 24
D 2- 28
```

Ví dụ 2: Mẫu thiết kế hướng đối tượng Composite

Xét khái niệm về tập tin và thư mục là quan hệ phân cấp dạng cây với thành phần xử lý giống nhau như tổng kích thước.

```
class Component
{
protected:
	string _name;
public:
	Component(const string& s)
	{
		_name = s;
	}
	virtual int GetSize() = 0;
	void Print(ostream& os)
	{
		os << _name << " (" << GetSize() << ")" << endl;
	}
};

class Directory : public Component
{
private:
	vector<Component*> _children;
public:
	Directory(const string& s) : Component(s) {}
	void AddChild(Component* c)
	{
		_children.push_back(c);
	}
	int GetSize()
	{
		int total = 0;
		for (int i = 0; i < _children.size(); i++)
		{
			total += _children[i]->GetSize();
		}
```

```
		return total;
	}
};

class File : public Component
{
private:
	int _size;
public:
	File(const string& s, const int& size) : Component(s)
	{
		_size = abs(size);
	}
	int GetSize()
	{
		return _size;
	}
};
```

Run với hàm main sau

```
int main()
{
	Directory* dRoot = new Directory("Dir root");
	Directory* dChild = new Directory("Dir child");
	dChild->AddChild(new File("File 1", 123));
	dChild->AddChild(new File("File 2", 444));
	dRoot->AddChild(dChild);
	dRoot->AddChild(new File("File 3", 7899));

	dRoot->Print(cout);

	cout << endl << endl;
	system("pause");
	return 0;
}
```

Ví dụ 3: Mẫu thiết kế hướng đối tượng Singleton

Giả sử cần cài đặt xử lý xuất kết quả ra 1 thiết bị xuất duy nhất, khi đó cần khống chế việc tạo nhiều hơn 1 instance.

```
class PrintConsoleDevice
{
private:
	ostream* os;
	PrintConsoleDevice() { os = &cout; }

public:
	void Print(const string& s)
	{
		*os << s;
	}
	static PrintConsoleDevice GetInstance()
	{
		static PrintConsoleDevice instance;
```

```
            return instance;
        }
};

int main()
{
        PrintConsoleDevice print = PrintConsoleDevice::GetInstance();
        print.Print("Su dung in lan 1\n");
        PrintConsoleDevice print2 = PrintConsoleDevice::GetInstance();
        print2.Print("Su dung in lan 2\n\n");

        system("pause");
        return 0;
}
```