HƯỚNG DẪN THỰC HÀNH
KIÊU DỮ LIỆU CẤU TRÚC - STRUCT

Tổng Quan

- Cấu trúc struct dùng để nhóm nhiều thành phần dữ liệu lại với nhau dưới một cái tên. Các thành phần này người ta gọi là thuộc tính, có kiểu dữ liệu có thể giống hoặc khác nhau.
- Cấu trúc struct được dùng để định nghĩa một kiểu dữ liệu mới.
- Trong C++, cú pháp để khai báo một cấu trúc như sau:

```
struct Ten_Struct{
      <kieu_du_lieu>  ten_thuoc_tinh_1;
      <kieu_du_lieu>  ten_thuoc_tinh_2;
      <kieu_du_lieu>  ten_thuoc_tinh_2;
      ...
};
```

Trong đó, Ten_Struct là tên của kiểu cấu trúc, do người khai báo tự đặt.

Ví dụ: struct sau định nghĩa một kiểu dữ liệu mới tên là `TraiCay`, gồm có 2 thuộc tính là:

- `trongLuong`: cho biết cân nặng của trái cây
- `donGia`: cho biết giá của trái cây đó.

```
struct TraiCay{
      float trongLuong;
      float donGia;
};
```

- Cú pháp để khai báo biến kiểu struct:

```
Ten_Struct  ten_bien;
```

Ví dụ:

```
TraiCay chuoi;
TraiCay xoai;
TraiCay sauRieng;
```

- Cú pháp để truy xuất đến các thuộc tính của cấu trúc:

```
Ten_bien_struct.ten_thuoc_tinh
```

Ví dụ:

```
chuoi.trongLuong = 1;  // nặng 1g
chuoi.donGia = 1000.0f; // có giá là 1000đ
```

Ví dụ:

```
// ví dụ về struct
#include <iostream>
#include <string>
#include <sstream>
using namespace std;

// Dùng struct định nghĩa kiểu dữ liệu phim
struct Phim{
	string tuaDe;
	int namPhatHanh;
};

void HienThiPhim (Phim p);

void main ()
{
	// Khai báo 2 biến vnPhim và usaPhim có kiểu là Phim
	Phim vnPhim, usaPhim;

	// Gán giá trị cho các thuộc tính
	vnPhim.tuaDe = "Hoa Co May";
	vnPhim.namPhatHanh = 2013;

	cout << "Nhap phim nuoc ngoai" << endl;
	cout << "Tua phim: ";
	cin >> usaPhim.tuaDe;

	cout << "Nam phat hanh: ";
	cin >> usaPhim.namPhatHanh;

	cout << "Phim Viet Nam la:" << endl;
	HienThiPhim(vnPhim);

	cout << "Phim nuoc ngoai yeu thich:" << endl;
```

```
	HienThiPhim(usaPhim);

}

void HienThiPhim(Phim p)
{
	cout << p.tuaDe;
	cout << " (" << p.namPhatHanh << ")" << endl;
}
```

Mảng các đối tượng cấu trúc

```
#include "stdafx.h"
#include <iostream>
#include <string>
using namespace std;

#define MAX 3

struct Phim {
  string tuaDe;
  int namPhatHanh;
};

void HienThiPhim (Phim p);

void main ()
{
	// Khai báo mảng tĩnh một chiều, mỗi phần tử có kiểu là Phim
	Phim films[MAX];
	for (int i = 0; i < MAX; i++)
	{
		cout << "Nhap tua phim: ";
		cin >> films[i].tuaDe;

		cout << "Nhap nam phat hanh: ";
		cin >> films[i].namPhatHanh;
	}

	cout << "Cac phim da nhap: " << endl;
	for (int i = 0; i < MAX; i++)
		HienThiPhim(films[i]);
}

void HienThiPhim (Phim p)
{
  cout << p.tuaDe;
  cout << " (" << p.namPhatHanh << ")\n";
```

```
}
```

Con trỏ & Cấu trúc

Giống các kiểu dữ liệu khác, cấu trúc có thể được trỏ đến bởi con trỏ có kiểu tương ứng. Xét ví dụ sau:

```
struct Phim {
  string tuaDe;
  int namPhatHanh;
};
```

Theo đó, có thể khai báo các biến sau:

```
Phim vnPhim;
Phim *pPhim;
```

Trong đó, vnPhim là một biến có kiểu cấu trúc là Phim và pPhim là một con trỏ, trỏ đến đối tượng có kiểu cấu trúc là Phim. Do đó, ta có thể thực hiện phép gán sau:

```
pPhim = &vnPhim;
```

Để truy xuất đến các thuộc tính của cấu trúc thông qua con trỏ, ta sử dụng dấu "→".

Ví dụ:

```
pPhim->tuaDe;
pPhim->namPhatHanh;
```

Cú pháp này tương đượng với

```
(*pPhim).tuaDe;
(*pPhim).namPhatHanh;
```

Ví dụ

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

struct Phim {
  string tuaDe;
  int namPhatHanh;
};

void HienThiPhim (Phim p);

void main ()
{
      Phim vnPhim;
      Phim *pVnPhim;
```

```
	pVnPhim = &vnPhim;

	Phim *pUsaPhim = new Phim;
	// Khai báo mảng tĩnh một chiều, mỗi phần tử có kiểu là Phim

	cout << "Nhap tua phim Viet Nam: ";
	cin >> pVnPhim->tuaDe;

	cout << "Nhap nam phat hanh: ";
	cin >> pVnPhim->namPhatHanh;

	cout << "Nhap tua phim nuoc ngoai: ";
	cin >> pUsaPhim->tuaDe;

	cout << "Nhap nam phat hanh: ";
	cin >> pUsaPhim->namPhatHanh;

	cout << "Cac phim da nhap: " << endl;
	HienThiPhim(vnPhim);
	HienThiPhim(*pUsaPhim);
}

void HienThiPhim (Phim p)
{
  cout << p.tuaDe;
  cout << " (" << p.namPhatHanh << ")\n";
}
```

Ví dụ: Mảng động các đối tượng cấu trúc

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

struct Phim {
  string tuaDe;
  int namPhatHanh;
};

void HienThiPhim (Phim p);

void main ()
{
	int n = 3;
	Phim *pPhim;
	pPhim = new Phim[n];
```

```
	for(int i = 0; i < n; i++)
	{
		cout << "Nhap tua phim: ";
		cin >> pPhim[i].tuaDe; // hoặc (pPhim + i)->tuaDe;

		cout << "Nhap nam phat hanh: ";
		cin >> pPhim[i].namPhatHanh;// hoặc (pPhim + i)-
>namPhatHanh;
	}

	for(int i = 0; i < n; i++)
	{
		HienThiPhim(pPhim[i]); // hoặc HienThiPhim(*(pPhim + i));
	}
}

void HienThiPhim (Phim p)
{
  cout << p.tuaDe;
  cout << " (" << p.namPhatHanh << ")\n";
}
```

Cấu trúc lồng nhau

Một cấu trúc có thể chứa các thuộc tính có kiểu cấu trúc.

```
struct Phim {
  string tuaDe;
  int namPhatHanh;
};

struct KhanGia {
	string ten;
	string email;
	Phim boPhimYeuThich;
};
```

Khai báo các đối tượng sau:

```
KhanGia an;
KhanGia *pKhanGia;
```

Cú pháp để truy xuất đến các thuộc tính của đối tượng:

```
an.ten;
an.boPhimYeuThich.namPhatHanh;

pKhanGia->ten;
pKhanGia->boPhimYeuThich.namPhatHanh;
```