Chương 5: Cấu trúc

5.1. Nhắc lại cách khai báo một kiểu dữ liệu mới (*typedef*):

a. Cú pháp:

typedef Kiểu_dữ_liệu_cũ Kiểu_dữ_liệu_mới;

b. Ý nghĩa:

Khi khai báo một kiểu dữ liệu mới bằng **typedef**, tức là

- **Kiểu_dữ_liệu_mới** là 1 kiểu dữ liệu như kiểu **Kiểu_dữ_liệu_cũ**
- Ta có thể sử dụng kiểu dữ liệu mới như những kiểu dữ liệu bình thường khác (sử dụng giống như kiểu dữ liệu int, float, double, char, …)

c. Ví dụ:

```
typedef  int  so_nguyen;        //Định nghĩa so_nguyen la kiểu dữ liệu mới
so_nguyen  i;                   //Sử dụng kiểu so_nguyen như kiểu dữ liệu bình
int  j;                         //thường khác
for (so_nguyen  t = 0; t<5; t++)
   cout<<"Hello world"<<endl;
```

5.2. Cách khai báo một kiểu dữ liệu mới bằng cú trúc:

a. Mục đích:

Trên thực tế, nhiều đối tượng có chứa nhiều thành phần riêng lẽ. Ví dụ, PHÂN SỐ có 2 thành phần là TỬ và MẪU, hoặc ĐIỂM thì có HOÀNH ĐỘ X và TUNG ĐỘ Y. Xu hướng lập trình hiện nay là gộp các thành phần riêng lẽ thành một kiểu dữ liệu mới, mà kiểu dữ liệu này sẽ chứa thông tin của các thành phần đó. Ví dụ, ta sẽ xây dựng kiểu dữ liệu PHÂNSỐ, với 2 trường (flied) là TỬ và MẪU, trong đó TỬ và MẪU là 2 trường kiểu số nguyên

b. <u>Cú pháp:</u>

i. **<u>Cách 1</u>**: dùng tyedef để định nghĩa kiểu mới

```
typedef   struct{
        kiểu_dữ_liệu    trường_1;
        kiểu_dữ_liệu    trường_2;
        .....
}tên_cấu_trúc;
```

ii. **<u>Cách 2:</u> không dùng** tyedef để định nghĩa kiểu mới

```
struct  tên_cấu_trúc{
        kiểu_dữ_liệu    trường_1;
        kiểu_dữ_liệu    trường_2;
        .....
};
```

khai báo tên cấu trúc đồng thời có thể khai báo cùng lúc các biến kiểu cấu trúc.

```
struct  tên_cấu_trúc{
        kiểu_dữ_liệu    trường_1;
        kiểu_dữ_liệu    trường_2;
        .....
}biến_1, biến_2,...;
```

<u>Lưu ý</u>:

- **Có dấu chấm phẩy (;) ở cuối khai báo cấu trúc**
- **Ta sử dụng kiểu cấu trúc như mọi kiểu bình thường khác**

c. **Ví dụ**:

Xây dựng kiểu cấu trúc PHÂN SỐ, với 2 trường là TỬ và MẪU kiểu số nguyên

Cách 1:

```
typedef struct
{
        int   tu;
        int   mau;
} PHANSO;
```

Cách 2:

```
struct  PHANSO
{
        int   tu;
        int   mau;
};
```

khi đó ta có thê sử dụng kiểu cấu trúc PHANSO như là một kiểu bình thường

```
PHANSO   ps;
```

c. **Cách truy cập vào các trường của cấu trúc**:

Ta dùng cú pháp sau để truy cập vào trường của cấu trúc

Tên_biến_kiểu_cấu_trúc**.**trường

(*lưu ý: có dấu chấm (.) ngăn cách giữa biến_kiểu_cấu_trúc và trường được gọi*)

Ví dụ, với cấu trúc PHANSO được khai báo như ở trên, ta có thể nhập và xuất các trường của biến ps có kiểu cấu trúc PHANSO như sau:

```
PHANSO ps;
cin>>ps.tu>>ps.mau;
cout<<ps.tu<<"/"<<ps.mau;
```

5.2. **Bài tập cấu trúc**

1) Viết chương trình nhập vào 2 phân số ps1, ps2
   - Tính tổng, hiệu, tích, nhân 2 phân số đó, kết quả trả về là phân số

   (Lưu ý: phải kiểm tra tính đúng đắn của dữ liệu nhập vào, ví dụ mẫu của phân số phải luôn khác không)
2) Viết chương trình nhập vào 3 điểm A, B, C trong hệ trục tọa độ Oxy, hãy:
   - Kiểm tra xem 3 điểm đó có thẳng hàng hay không
   - Tính khoảng cách từ A đến B, A đến C, và B đến C
   - Nếu A,B,C không thẳng hàng hãy tính diện tích, chu vi của tam giác ABC
3) Viết chương trình quản lý điểm của sinh viên, với
   - Số lượng sinh viên là n, với n nhập từ bàn phím
   - Thông tin về sinh viên gồm:
     - Họ và tên
     - Ngày tháng năm sinh
     - MSSV
     - Điểm Toán, Lý, Hoá
     - Điểm trung bình