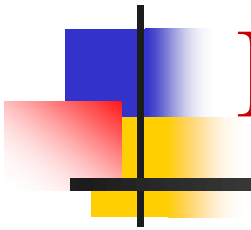

DỮ LIỆU KIỂU CẤU TRÚC

- Khái niệm và khai báo cấu trúc
- Truy nhập vào các trường của cấu trúc
- Sử dụng mảng cấu trúc
- Truyền tham số cấu trúc cho hàm
- Con trỏ cấu trúc

3.1. Khái niệm cấu trúc

- Một cấu trúc bao gồm các thành phần dữ liệu, không nhất thiết cùng kiểu, được nhóm lại với nhau.

Định nghĩa cấu trúc

- Cú pháp định nghĩa câu trúc:

```
struct <Tên cấu trúc>
{
   Khai báo các thành phần dữ liệu;
};
```

- Ví dụ:

```
struct Sach{
    char TenSach [25];
    char TacGia [20];
    int NamXB;
    float Gia;
};
```

Định nghĩa cấu trúc

- Một định định nghĩa cấu trúc:
  - Tạo ra kiểu dữ liệu mới.
  - Cho phép sử dụng để khai báo các biến kiểu cấu trúc
- Các biến trong cấu trúc được gọi là các phần tử của cấu trúc hay thành phần của cấu trúc.

Khai báo biến kiểu cấu trúc

- Khi một cấu trúc đã được định nghĩa, chúng ta có thể khai báo một hoặc nhiều biến kiểu này

```
struct Sach{
        char TenSach[25];
        char TacGia[20];
        int NamXB;
        float Gia;
} s1, s2;
```

Cách khác

```
struct Sach s1, s2;
```

hoặc

```
struct Sach s1;
struct Sach s2;
```

Từ khóa typedef

- Một kiểu dữ liệu có thể được định nghĩa bằng cách sử dụng từ khóa **typedef**
- Nó không tạo ra một kiểu dữ liệu mới, mà định nghĩa một tên mới cho một kiểu đã có.
- Cú pháp:

typedef <Kiểu dữ liệu> <Tên mới>;

Ví dụ:

```
typedef int Int32;
```

Từ khóa typedef

- Sử dụng từ khóa typedef trong định nghĩa cấu trúc:

```
typedef struct
{
   Khai báo các thành phần dữ liệu;
} <Tên cấu trúc>;
```

```
typedef struct {
    char TenSach [25];
    char TacGia [20];
    int NamXB;
    float Gia;
}Sach;
```

Từ khóa typedef

- Các cấu trúc được định nghĩa với từ khóa typedef :

```
Struct SoPhuc
{
   int thuc;
   int ao;
};

struct SoPhuc sp;
```

```
typedef struct
{
   int thuc;
   int ao;
}SoPhuc;

SoPhuc sp;
```

3.2 Truy cập các trường cấu trúc

- Các phần tử (trường) của cấu trúc được truy cập thông qua việc sử dụng **toán tử chấm** (.).
- Cú pháp:

<Tên biến cấu trúc>.<Tên thành phần>

Ví dụ:

```
printf("%s", s1.TenSach);
```

Khởi tạo biến cấu trúc

- Các biến kiểu cấu trúc có thể được khởi tạo tại thời điểm khai báo

```
struct NhanVien {
        int MaNV;
        char TenNV [20];
};
```

- Biến **nv1** có kiểu **NhanVien** có thể được khai báo và khởi tạo như sau:

```
Struct NhanVien nv1 = {100,"John"};
```

Lệnh gán biến cấu trúc

- Có thể sử dụng câu lệnh gán đơn giản để gán giá trị của một biến cấu trúc cho một biến khác ***có cùng kiểu***

s2 = s1;

- Trường hợp không thể dùng câu lệnh gán trực tiếp, thì có thể sử dụng hàm tạo sẵn **memcpy()**

memcpy (void * destn, void *source, size_t size);

- Ví dụ

memcpy (&s2, &s1, sizeof(struct Sach));

Cấu trúc lồng nhau

- Một cấu trúc có thể lồng trong một cấu trúc khác. Tuy nhiên, một cấu trúc không thể lồng trong chính nó.

```
struct Date{
  int Ngay, Thang, Nam;
};

struct SinhVien    {
   int MaSV;
   char HoTen[30];
   struct Date NgaySinh;
}sv;
```

Cấu trúc lồng nhau

- Truy cập vào các phần tử của cấu trúc này tương tự như với cấu trúc bình thường khác.

```
sv.MaSV
```

- Truy cập vào phần tử của cấu trúc là một phần của cấu trúc khác:

```
sv.NgaySinh.Ngay
```

Bài tập

- Ví dụ: ViduCauTruc
- Viết chương trình thực hiện yêu cầu:
  - Khai báo cấu trúc SinhVien (MaSV, TenLop, HoTen, Diem, ngaysinh (Ngay, Thang, Nam));
  - Khai báo và nhập dữ liệu cho sinh viên từ bàn phím.
  - Hiển thị các giá trị đã nhập ra màn hình theo cột.

3.3. Mảng cấu trúc

- Một kiểu cấu trúc phải được định nghĩa trước, sau đó khai báo biến mảng có kiểu cấu trúc;
- Cú pháp

  struct <Tên cấu trúc> <Tên mảng> [số phần tử]

  Ví dụ: `struct Sach SAry[50];`
- Để truy cập vào thành phần **TenSach** của phần tử thứ tư của mảng **SAry**:

  `SAry[4].TenSach;`

Bài tập

- Ví dụ: ViduCauTrucMang
- Viết chương trình thực hiện các yêu cầu:
  - Khai báo cấu trúc SinhVien (MaSV, TenLop, HoTen, Diem, ngaysinh (Ngay, Thang, Nam));
  - Khai báo và nhập dữ liệu cho mảng n sinh viên từ bàn phím.
  - Hiển thị các giá trị đã nhập ra màn hình theo cột.

3.4. Con trỏ cấu trúc

- Cú pháp khai báo con trỏ cấu trúc

  `struct` <Tên cấu trúc> **`*`** <Tên biến trỏ>

  Ví dụ:

```
struct Sach *ptr;
```

- Toán tử -> được dùng để truy cập vào các phần tử của một cấu trúc sử dụng một con trỏ

```
struct Sach s;
ptr = &s;
printf("%s",ptr->TacGia);
```

3.5. Truyền tham số cho hàm

- Truyền tham trị
  - Khai báo nguyên mẫu hàm

    func (struct <Tên cấu trúc> <Tên biến >);

    Ví dụ:

    ```
    void HienThi (struct Sach s);
    ```
  - Gọi hàm

    <Tên hàm> (<Tên biến cấu trúc>)

    ```
    struct Sach s;
    HienThi(s);
    ```

3.5. Truyền tham số cho hàm

- Truyền tham chiếu trực tiếp qua địa chỉ
  - Khai báo nguyên mẫu hàm

    func (struct <Tên cấu trúc> & <Tên biến >);

    Ví dụ:

    ```
    void Nhap(struct Sach &s);
    ```

  - Gọi hàm

    <Tên hàm> (<Tên biến cấu trúc>)

    ```
    struct Sach s;
    Nhap(s);
    ```

3.5. Truyền tham số cho hàm

- Truyền tham chiếu gián tiếp qua con trỏ
  - Khai báo nguyên mẫu hàm

    func (struct <Tên cấu trúc> * <Tên biến >);

    Ví dụ:

    ```
    void Nhap(struct Sach *s);
    ```

  - Gọi hàm

    <Tên hàm> (<Tên biến trỏ>)

    Hoặc

    <Tên hàm> (&<Tên biến cấu trúc>)

Bài tập

- Ví dụ: ViduCauTrucMangHam
- Viết chương trình dưới dạng hàm thực hiện các yêu cầu:
  - Khai báo cấu trúc SinhVien (MaSV, TenLop, HoTen, Diem, ngaysinh (Ngay, Thang, Nam));
  - Hàm nhập dữ liệu cho mảng n sinh viên từ bàn phím.
  - Hàm hiển thị các giá trị đã nhập ra màn hình theo cột.
  - Nhập vào một giá trị x (float) bất kỳ, hàm đếm số sinh viên có điểm trung bình >= x.