

# Kiểu cấu trúc - STRUCT

BUỔI 11



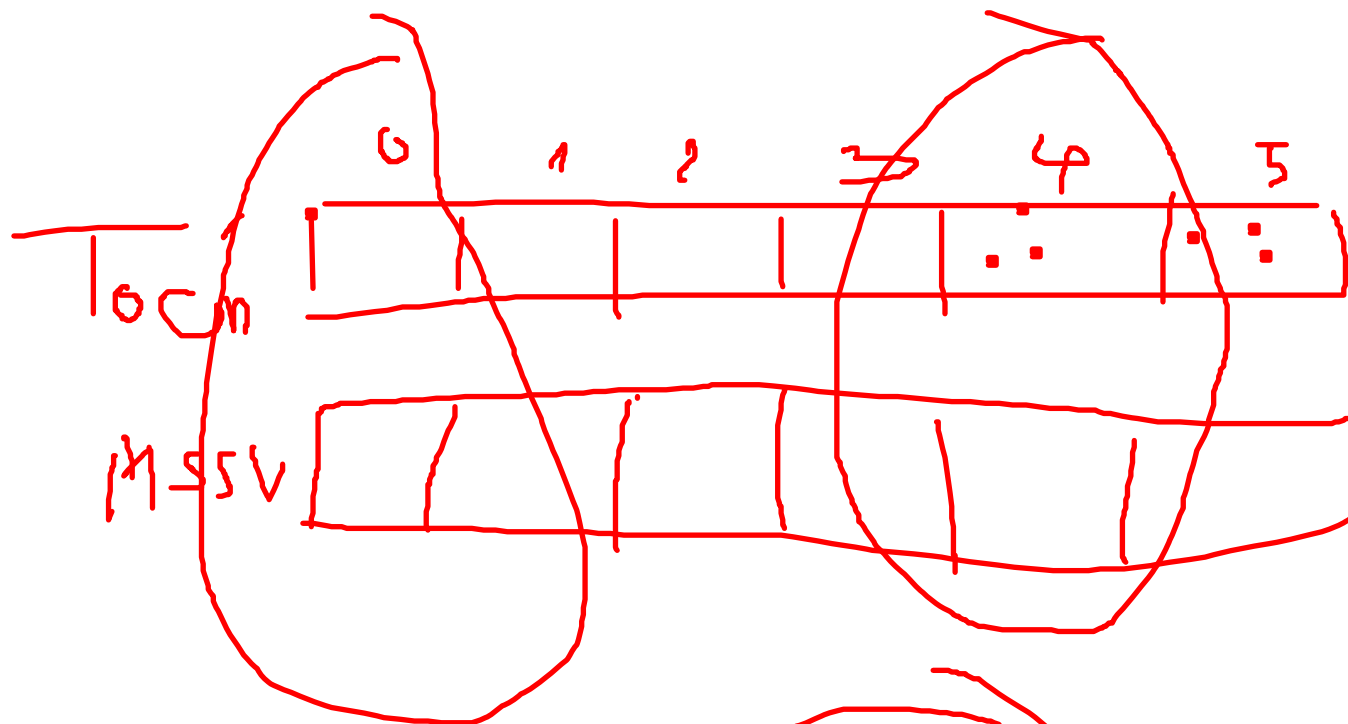
- Khái niệm kiểu cấu trúc (struct)
- Khai báo và truy xuất kiểu cấu trúc
- Bài tập

# Đặt vấn đề



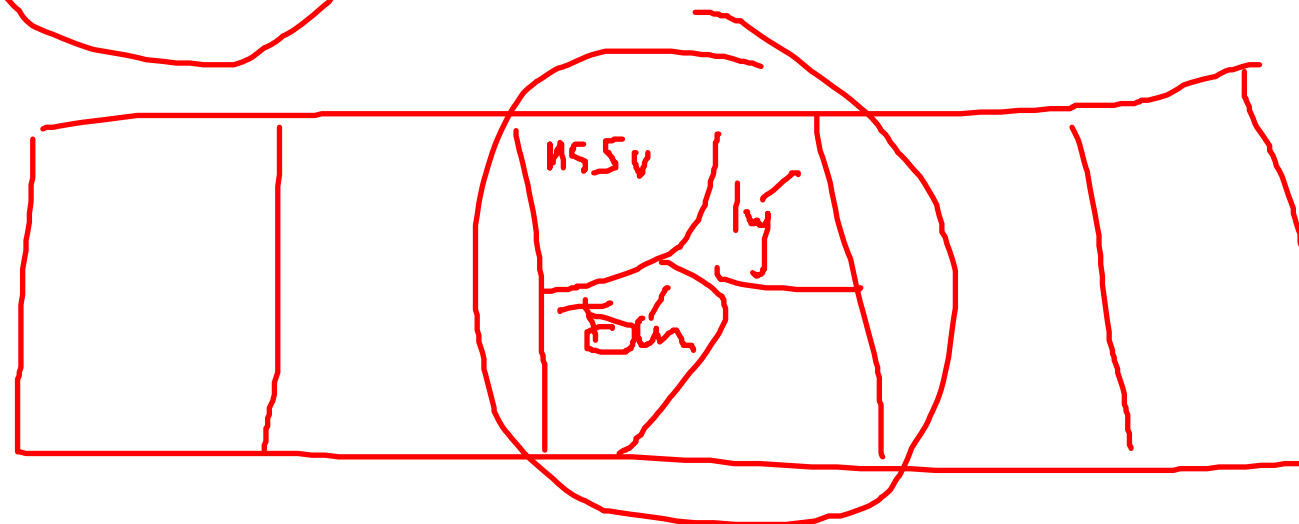
- Thông tin 1 sinh viên (SV)

- MSSV: kiểu chuỗi
- Tên SV: kiểu chuỗi
- Ngày tháng năm sinh: kiểu chuỗi
- Giới tính: ký tự
- Điểm toán, lý, hóa: số thực



- Yêu cầu

- Lưu thông tin cho N sinh viên ?
- Truyền thông tin N sinh viên vào một hàm ?





- Khai báo các biến để lưu trữ 1 SV
  - `char mssv[7];` // “0012078”
  - `char hoten[30];` // “Nguyen Van A”
  - `char ntns[8];` // “29/12/82”
  - `char phai;` // ‘y’ ⇔ Nam, ‘n’ ⇔ Nữ
  - `float toan, ly, hoa;` // 8.5 9.0 10.0
- Truyền thông tin 1 SV cho hàm
  - `void xuat(char mssv[], char hoten[], char ntns[], char phai, float toan, float ly, float hoa);`



- Nhận xét

- Đặt tên biến khó khăn và khó quản lý
- Truyền tham số cho hàm quá nhiều
- Tìm kiếm, sắp xếp, sao chép,... khó khăn
- Tốn nhiều bộ nhớ
- ...

- Ý tưởng

- Gom những thông tin của cùng 1 SV thành một kiểu dữ liệu mới => Kiểu **struct**

# Khai báo kiểu cấu trúc



- Cú pháp

```
▪ struct <tên kiểu cấu trúc>  
{  
    <kiểu dữ liệu> <tên thành phần 1>;  
    ...  
    <kiểu dữ liệu> <tên thành phần n>;  
};
```

- Ví dụ

```
struct DIEM  
{  
    int x;  
    int y;  
};
```

# Khai báo biến cấu trúc



- Cú pháp không tường minh

```
struct <tên kiểu cấu trúc>
{
    <kiểu dữ liệu> <tên thành phần 1>;
    ...
    <kiểu dữ liệu> <tên thành phần n>;
};
struct <tên kiểu cấu trúc> <tên biến>;
```

- Ví dụ

```
struct DIEM
{
    int x;
    int y;
};
struct DIEM diem1, diem2; // C++ có thể bỏ struct
```

# Sử dụng typedef



- Cú pháp

```
typedef struct
{
    <kiểu dữ liệu> <tên thành phần 1>;
    ...
    <kiểu dữ liệu> <tên thành phần n>;
} <tên kiểu cấu trúc>;
<tên kiểu cấu trúc> <tên biến>;
```

- Ví dụ

```
typedef struct DIEM
{
    int x;
    int y;
};
struct DIEM diem1, diem2; // C++ có thể bỏ struct
```



# Khởi tạo cho biến cấu trúc



- Cú pháp tường minh

```
struct <tên kiểu cấu trúc>
{
    <kiểu dữ liệu> <tên thành phần 1>;
    ...
    <kiểu dữ liệu> <tên thành phần n>;
} <tên biến> = {<giá trị 1>, ..., <giá trị n>;};
```

- Ví dụ

```
struct DIEM
{
    int x;
    int y;
} diem1 = {2912, 1706}, diem2;
```



- Đặc điểm
  - Không thể truy xuất trực tiếp
  - Thông qua toán tử thành phần cấu trúc hay còn gọi là toán tử chấm (dot operation)

- Ví dụ

<tên biến cấu trúc>.<tên thành phần>

```
struct DIEM
{
    int x;
    int y;
} diem1;
cout << diem1.x << diem1.y;
```

# Gán dữ liệu kiểu cấu trúc



- Có 2 cách

```
<biến cấu trúc đích> = <biến cấu trúc nguồn>;
```

```
<biến cấu trúc đích>.<tên thành phần> = <giá trị>;
```

- Ví dụ

```
struct DIEM
{
    int x, y;
} diem1 = {2912, 1706}, diem2;
...
diem2 = diem1;
diem2.x = diem1.x;
diem2.y = diem1.y * 2;
```

# Cấu trúc phức tạp

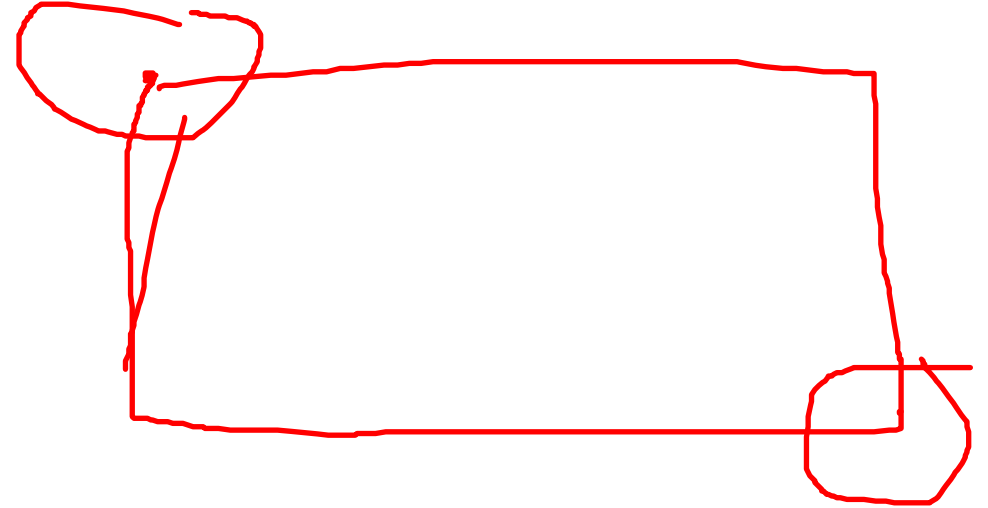


- Thành phần của cấu trúc là cấu trúc khác

```
struct DIEM
{
    int x;
    int y;
};

struct HINHCHUNHAT
{
    struct DIEM traitren;
    struct DIEM phaiduoi;
} hcn1;

...
hcn1.traitren.x = 2912;
hcn1.traitren.y = 1706;
```



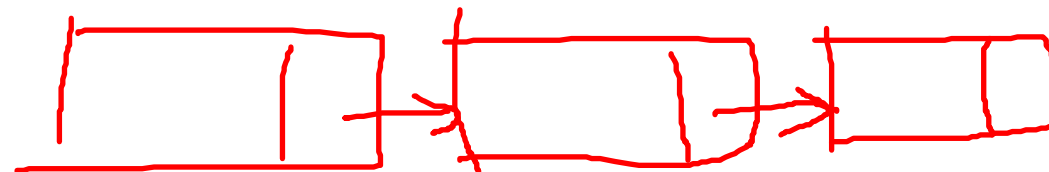
# Cấu trúc phức tạp



- Cấu trúc đệ quy (tự trỏ)

```
struct PERSON
{
    char hoten[30];
    struct PERSON *father, *mother;
};
```

```
struct NODE
{
    int value;
    struct NODE *pNext;
};
```





- Kiểu cấu trúc được định nghĩa để làm khuôn dạng còn biến cấu trúc được khai báo để sử dụng khuôn dạng đã định nghĩa.
- Trong C++, có thể bỏ từ khóa struct khi khai báo biến (hoặc sử dụng typedef)
- Khi nhập các biến kiểu số thực trong cấu trúc phải nhập thông qua một biến trung gian.

```
struct DIEM { float x, y;} d1;  
float temp;  
cin >> temp;  
d1.x = temp;
```



- Mảng cấu trúc
  - Tương tự như mảng với kiểu dữ liệu cơ sở (char, int, float, ...)

```
struct DIEM
{
    int x;
    int y;
};
```

```
DIEM mang1[20];
DIEM mang2[10] = {{3, 2}, {4, 4}, {2, 7}};
```

# Kích thước cấu trúc



```
struct B1
```

```
{
```

```
    int a;
```

```
    double c;
```

```
    int b;
```

```
};
```

```
Sizeof (B1) = ?? 24
```

```
struct B2
```

```
{
```

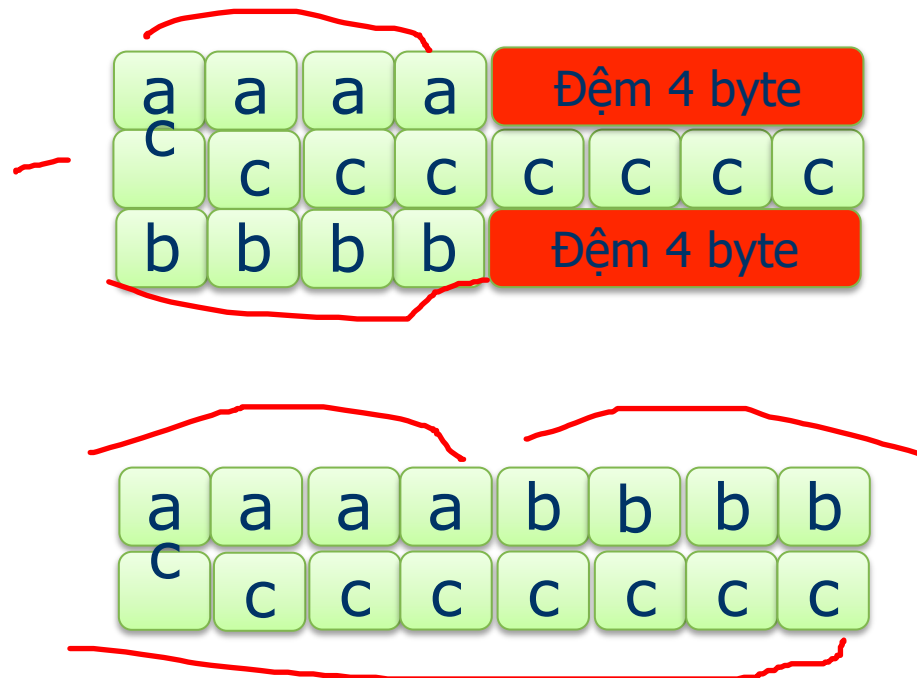
```
    int a;
```

```
    int b;
```

```
    double c;
```

```
};
```

```
Sizeof (B2) = ?? 16
```



Sự khác biệt đến từ **thứ tự khai báo các biến** và **biên kích thước** (tính theo byte) của cấu trúc.  
**Biên mặc định** của VC++ là 8 byte.





- Tối ưu kích thước cấu trúc dựa trên thứ tự các biến (tối ưu cục bộ trên cấu trúc).
- Hoặc tối ưu biên cho cấu trúc (alignment of struct). Ví dụ trên nếu thay đổi biên cấu trúc thành 1 hoặc 4 thì **sizeof(B1) = 16**.
- Điều chỉnh biên cấu trúc: **Project settings → Compile Option C/C++ → Code Generation → Structure Alignment**.
- Dễ dàng điều chỉnh biên để tối ưu. Hay biên cấu trúc càng nhỏ càng giúp giảm vùng đệm thì càng tốt ??
  - Không. Biên nhỏ giúp giảm kích thước của cấu trúc nhưng làm tăng thời gian xử lý của tác vụ memory allocator ← Cần điều phối thích hợp giữa kích thước cấu trúc và tốc độ xử lý.
  - Chương trình dùng nhiều cấu trúc có thành phần khác nhau điều chỉnh biên tốt nhất sẽ khó khăn
  - Ưu tiên: tối ưu bằng cách khai báo thứ tự các thành phần cấu trúc phù hợp với biên cấu trúc.

# Truyền cấu trúc cho hàm



- Giống như truyền kiểu dữ liệu cơ sở
  - Tham trị (không thay đổi sau khi kết thúc hàm)
  - Tham chiếu
  - Con trỏ

- Ví dụ

```
struct DIEM
{
    int x, y;
};
void xuat1(int x, int y) { ... };
void xuat2(DIEM diem) { ... };
void xuat3(DIEM &diem) { ... };
void xuat4(DIEM *diem) { ... };
```

# Bài tập minh họa



1. Xây dựng cấu trúc sinh viên: tên, mssv, lớp
2. Nhập, xuất cho danh sách sinh viên
3. Tìm sinh viên theo tên



## Tạo cấu trúc sinh viên

```
typedef struct SinhVien
{
    char ten[50];
    char mssv[10];
    char lop[5];
}SV;
```

## Nhập danh sách sinh viên

```
void NhapSinhVien(SV dssv[], int thutu)
{
    fflush(stdin);
    printf ("Nhap ten sinh vien : ");
    gets(dssv[thutu].ten);
    fflush(stdin);
    printf ("Nhap ma so sinh vien : ");
    gets(dssv[thutu].mssv);
    fflush(stdin);
    printf ("Nhap lop cua sinh vien : ");
    gets(dssv[thutu].lop);
}
```



## Xuất danh sách sinh viên

```
void XuatSinhVien(SV dssv[], int thutu)
{
    printf ("Ten sinh vien : ");
    puts(dssv[thutu].ten);
    printf ("Ma so sinh vien : ");
    puts(dssv[thutu].mssv);
    printf ("Lop cua sinh vien : ");
    puts(dssv[thutu].lop);
}
```

## Tìm sinh viên theo tên

```
int TimSVTheoTen(SV dssv[], int soluongsv, char
ten[])
{
    int vitritimthay = -1;
    for ( int i = 0; i < soluongsv; i++ )
    {
        if (strcmp(dssv[i].ten, ten) == 0)
        {
            vitritimthay = i;
            break;
        }
    }
    return vitritimthay;
}
```



1. Khai báo kiểu dữ liệu Đơn thức, nhập/xuất đơn thức, tính tổng/hiệu/tích/thương hai đơn thức, tính giá trị đơn thức, tính đạo hàm cấp 1 của đơn thức, ...
2. Khai báo kiểu dữ liệu điểm trong mặt phẳng Oxy, nhập/xuất tọa độ điểm/mảng điểm, Tính khoảng cách giữa hai điểm, tìm 1 điểm trong mảng gần/ xa gốc tọa độ nhất, ...
3. Hãy khai báo kiểu dữ liệu để biểu diễn thông tin của 1 tỉnh gồm mã tỉnh, tên tỉnh, dân số, diện tích; nhập xuất thông tin 1 tỉnh và danh sách tỉnh; xuất tỉnh có dân số lớn hơn 1 triệu, tìm tỉnh có diện tích lớn nhất, ...