

BỘ CÔNG THƯƠNG
ĐẠI HỌC CÔNG NGHIỆP TP. HỒ CHÍ MINH



Bài giảng

NGÔN NGỮ LẬP TRÌNH C
THE C PROGRAMMING LANGUAGE

Lecturer : Le Ngoc Tran, PhD

Email : lengoctran@iuh.edu.vn

CHƯƠNG 8

CẤU TRÚC (STRUCT) VÀ UNION

8.1. Tại sao phải dùng Struct?

- ☐ Lưu 100 lần giá trị **tuổi** kiểu **int**
- ☐ Lưu 100 lần giá trị **điểm** kiểu **float**
- ☐ Lưu 100 lần giá trị **tên** kiểu **char**
- ☐ Lưu 100 lần sinh viên (**Họ và tên, giới tính, địa chỉ, email, tuổi, điểm trung bình, số điện thoại**)??

➡ **Có thể dùng mảng (array) để lưu thông tin sinh viên hay?**

8.1. Tại sao phải dùng Struct?

- ❑ Để lưu được 100 sinh viên, chúng ta cần định nghĩa là *1 kiểu dữ liệu mới*, kiểu dữ liệu **Sinh viên**
- ❑ Ví dụ định nghĩa cấu trúc của **SinhViên**.

```
struct SinhVien {  
    int MaSV;  
    char ten[50];  
    float diem;  
};
```

8.2. Định nghĩa cấu trúc (**struct**)

- ❑ Cấu trúc (structure) là một kiểu dữ liệu người dùng tự định nghĩa cho phép lưu trữ **các loại phần tử có kiểu dữ liệu khác nhau.**
 - ❖ Mỗi phần tử được gọi là một thành viên (member)
- ❑ Từ khóa **struct** được sử dụng để xác định cấu trúc

8.2. Định nghĩa cấu trúc (**struct**)

- ❑ Cú pháp để định nghĩa cấu trúc trong c.

```
struct structure_name{  
    data_type member1;  
    data_type member2;  
    ....  
    data_type memberN;  
};
```



- ❑ Ví dụ:

```
struct SinhVien{  
    int MaSV;  
    char ten[50];  
    float diem;  
}
```

8.2. Định nghĩa cấu trúc (struct)

❑ Có hai cách để khai báo biến cấu trúc:

- ❖ Sử dụng từ khóa struct trong hàm **main()**
- ❖ Khai báo biến cấu trúc tại thời điểm định nghĩa cấu trúc

```
struct sinhvien {  
    int MaSV;  
    char ten[50];  
    float diem;  
};  
void main() {  
    struct sinhvien sv1, sv2;  
}
```

```
struct sinhvien {  
    int MaSV;  
    char ten[50];  
    float diem;  
} sv1, sv2;
```

8.2. Định nghĩa cấu trúc dùng **typedef**

- ❑ Khi định nghĩa cấu trúc có thể dùng **typedef**
- ❑ Cú pháp:

Typedef struct **structure_name**{

data_type member1;

data_type member2;

....

data_type memberN;

}Type_Name;

```
typedef struct sinhvien {  
    int MaSV;  
    char ten[50];  
    float diem;  
} SinhVien;
```


8.3. Khai báo biến cấu trúc định nghĩa với **typedef**

- ❑ Dùng tên kiểu khi khai báo
- ❑ Ví dụ:

```
typedef struct sinhvien {  
    int MaSV;  
    char ten[50];  
    float diem;  
} SinhVien;  
  
void main() {  
    SinhVien sv1, sv2;  
}
```

8.4. Gán dữ liệu kiểu cấu trúc

□ Có hai 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.x = 1008;
diem2.y = 4322;
```

Ví dụ: Khai báo cấu trúc

```
int main(int argc, char const *argv[])
{
    struct SinhVien{
        int maSV;
        char ten[50];
    };

    struct SinhVien sv1, sv2;

    typedef struct DiemStruct{
        int x, y;
    } Diem;

    Diem p1, p2;

    struct NhanVien{
        int maNV;
        char ten[50];
    } nv1, nv2;

    return 0;
}
```

8.5. Truy cập thành viên cho cấu trúc

- ❑ Không thể truy cập trực tiếp.
- ❑ Thông qua toán tử thành phần cấu trúc “.” hay còn gọi là toán tử **chấm** (dot operation)
- ❑ Cú pháp:

< tên biến cấu trúc>.<tên thành phần>
- ❑ Ví dụ:

```
struct sinhvien {  
    int MaSV;  
    char ten[50];  
    float diem;  
} sv1;  
printf("MaSV = %d, Ten = %s, Diem = %f" ,sv1.MaSV, sv1.ten, sv1.diem);
```

8.6. Gán biến cấu trúc

❑ Toán tử gán “=” có thể dùng để gán biến cấu trúc

```
struct sinhvien {  
    int MaSV;  
    char ten[50];  
    float diem;  
};  
void main() {  
    struct sinhvien sv1, sv2;  
  
    sv1 = sv2;  
}
```

8.6. Gán biến cấu trúc

- ❑ Trong trường hợp không thể gán trực tiếp, hàm **memcpy()** có thể được dùng:
- ❑ Cú pháp `memcpy (char * destn, char &source, int nbytes);`
- ❑ Ví dụ: `memcpy (&sv1, &sv2, sizeof(struct sinhvien));`

8.7. Mạng cấu trúc

- ❑ Cấu trúc (**Structure**) giống như khuôn làm bánh
- ❑ Biến cấu trúc: Nơi chứa những cái bánh (được làm từ khuôn bánh)
- ❑ Mạng cấu trúc: Tủ chứa bánh (Có thể chứa nhiều hay ít tùy khai báo)

8.7. Mạng cấu trúc

- ❑ Có thể khai báo và sử dụng mảng của cấu trúc (Structure) để lưu trữ nhiều thông tin của các loại dữ liệu khác nhau.
- ❑ Ví dụ: Cấu trúc với mảng lưu trữ thông tin của 5 sinh viên và in các phần tử của nó ra màn hình

```
struct student {  
    int id;  
    char name[10];  
};
```

```
int i;  
struct student st[5];  
printf("Nhap thong tin cho 5 sinh vien: \n");  
for (i = 0; i < 5; i++) {  
    printf("Nhap id: ");  
    scanf("%d", &st[i].id);  
    printf("Nhap name: ");  
    scanf("%s", &st[i].name);  
}  
printf("Danh sach sinh vien: \n");  
for (i = 0; i < 5; i++) {  
    printf("Id: %d, Name: %s\n", st[i].id, st[i].name);  
}
```


Ví dụ:Nhập – xuất mảng cấu trúc

```
{
    struct SinhVien
    {
        int maSV;
        char ten[50];
    };

    struct SinhVien sv[5];

    for (int i = 0; i < 5; i++)
    {
        printf("Nhap ma sv[%d]: ", i);
        scanf("%d", &(sv[i].maSV));
        printf("Nhap ten sv[%d]: ", i);
        fflush(stdin);
        scanf("%[^\n]", sv[i].ten);
    }

    for (int i = 0; i < 5; i++)
    {
        printf("Ma SV: %d - Ten SV: %s \n", sv[i].maSV, sv[i].ten);
    }

    return 0;
}
```

```
PS C:\MyData\Demonstration\PolyCLang
Nhap ma sv[0]: 1
Nhap ten sv[0]: Anh
Nhap ma sv[1]: 2
Nhap ten sv[1]: Minh
Nhap ma sv[2]: 3
Nhap ten sv[2]: Hung
Nhap ma sv[3]: 4
Nhap ten sv[3]: Trang
Nhap ma sv[4]: 5
Nhap ten sv[4]: Ha
Ma SV: 1 - Ten SV: Anh
Ma SV: 2 - Ten SV: Minh
Ma SV: 3 - Ten SV: Hung
Ma SV: 4 - Ten SV: Trang
Ma SV: 5 - Ten SV: Ha
```

8.8. Cấu trúc lồng nhau

- ❑ Có thể sử dụng **structure** bên trong **structure** khác, nó được biết đến như structure lồng nhau trong C.
- ❑ Có 2 cách để định nghĩa cấu trúc lồng nhau trong C:
 - ❖ Theo cấu trúc riêng biệt
 - ❖ Theo cấu trúc nhúng.

8.8. Cấu trúc lồng nhau

- ❑ Ví dụ: chúng ta tạo ra cấu trúc và cấu trúc phụ thuộc được sử dụng bên trong cấu trúc chính như một thành viên.

Trong ví dụ trên, cấu trúc **ngaysinh** được sử dụng như một thành viên của cấu trúc **sinhvien**.

```
struct Date {  
    int ngay;  
    int thang;  
    int nam;  
};  
struct sinhvien {  
    int MaSV;  
    char ten[20];  
    struct Date ngaysinh;  
} sv1;
```

Ví dụ cấu trúc lồng nhau:

nested-struct.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int main(int argc, char *argv[])
6  {
7      struct Date{
8          int day, month, year;
9      };
10     struct Student{
11         int studentId;
12         char name[50];
13         struct Date dob;
14     };
15     struct Student sd1;
16     sd1.studentId = 1;
17     strcpy(sd1.name, "AnhNN");
18     sd1.dob.day = 2;
19     sd1.dob.month = 3;
20     sd1.dob.year = 2021;
21     printf("Student ID: %d \n", sd1.studentId);
22
23     return 0;
24 }
```

8.9. Cấu trúc lồng nhau theo cấu trúc nhúng

❑ Cấu trúc nhúng là việc định nghĩa cấu trúc bên trong một cấu trúc khác.


❑ Ví dụ:

```
struct sinhvien {  
    int MaSV;  
    char ten[20];  
    struct Date {  
        int ngay;  
        int thang;  
        int nam;  
    } ngaysinh;  
} sv1;
```

8.10. Truy cập cấu trúc lồng nhau

- ❑ Bạn có thể truy cập các thành viên của cấu trúc lồng nhau bởi `Outer_Structure.Nested_Structure.member` như dưới đây.
- ❑ Ví dụ:

```
struct sinhvien {  
    int MaSV;  
    char ten[20];  
    struct Date {  
        int ngay;  
        int thang;  
        int nam;  
    } ngaysinh;  
} sv1;
```



```
sv1.ngaysinh.ngay  
sv1.ngaysinh.thang  
sv1.ngaysinh.nam
```

Ví dụ cấu trúc lồng nhau:

```
struct sinhvien {  
    int MaSV;  
    char ten[20];  
    struct Date {  
        int ngay;  
        int thang;  
        int nam;  
    } ngaysinh;  
} sv1;
```

```
sv1.MaSV = 101;  
strcpy(sv1.ten, "Phong Tran"); // chuyển đổi chuỗi thành mảng  
char  
sv1.ngaysinh.ngay = 10;  
sv1.ngaysinh.thang = 11;  
sv1.ngaysinh.nam = 1998;  
  
// hiển thị thông tin sinh viên ra màn hình  
printf("Ma so sinh vien: %d\n", sv1.MaSV);  
printf("Ten sinh vien: %s\n", sv1.ten);  
printf("Ngày sinh (dd/mm/yyyy): %d/%d/%d\n",  
sv1.ngaysinh.ngay, sv1.ngaysinh.thang, sv1.ngaysinh.nam);
```

```
Ma so sinh vien: 101  
Ten sinh vien: Phong Tran  
Ngày sinh sinh vien (dd/mm/yyyy): 10/11/1998  
Press any key to continue . . .
```

8.11. Khai báo union

- ❑ **Union** trong C là kiểu dữ liệu do người dùng định nghĩa được sử dụng để chứa các loại phần tử khác nhau.
- ❑ Được khai báo và sử dụng như cấu trúc.
- ❑ Các thành phần của **union** có chung địa chỉ đầu (nằm chồng lên nhau trong bộ nhớ).

8.11. Khai báo union

□ Cú pháp:

union <ten kieu union>

{

<kiểu dữ liệu> <tên thành phần 1>;

....

<kiểu dữ liệu> <tên thành phần 2>;

}

□ Ví dụ:

```
union SinhVien{  
    char ten[100];  
    int tuoi, diem;  
};
```

```
union date
```

```
{
```

```
    int d;
```

```
    int m;
```

```
    int y;
```

```
};
```

```
void main()
```

```
{
```

```
    date dat;
```

```
    printf("\nSize of union: %d", sizeof(date));
```

```
    dat.d = 24;
```

```
    printf("\ndate = %d", dat.d);
```

```
    dat.m = 9;
```

```
    printf("\nmonth = %d", dat.m);
```

```
    dat.y = 2019;
```

```
    printf("\nyear = %d", dat.y);
```

```
    getch();
```

```
}
```

union.c

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int main(int argc, char *argv[])
6  {
7      union Student{
8          int studentId;
9          char name[50];
10     };
11     union Student st1;
12
13     st1.studentId = 9;
14     strcpy(st1.name, "AnhNN");
15     printf("Student ID: %d - name: %s \n", st1.studentId, st1.name);
16
17     strcpy(st1.name, "Dung");
18     st1.studentId = 9;
19     printf("Student ID: %d - name: %s \n", st1.studentId, st1.name);
20
21     return 0;
22 }
```

8.12. So sánh **struct** và **union**

struct

```
struct SinhVien{  
    char ten[100];  
    int tuoi, diem;  
};  
  
int main(){  
    struct SinhVien sv;  
    printf("Kich thuoc cua cau truc struct la: %d  
byte",sizeof(sv));  
    return 0;  
}
```

**Kich thuoc cua cau truc struct
la:108 byte**

union

```
union SinhVien{  
    char ten[100];  
    int tuoi, diem;  
};  
  
int main(){  
    union SinhVien sv;  
    printf("Kich thuoc cua cau truc union la: %d  
byte",sizeof(sv));  
    return 0;  
}
```

**Kich thuoc cua cau truc union
la:100 byte**

8.12. So sánh **struct** và **union**

struct

- ❑ Size của **struct** ít nhất bằng tổng size của các thành phần của struct.
- ❑ Tại cùng 1 thời điểm run-time, có thể truy cập vào tất cả các thành phần của struct.

union

- ❑ Size của **union** bằng size của thành phần có size lớn nhất trong union.
- ❑ Tại cùng 1 thời điểm run-time, chỉ có thể truy cập 1 thành phần của union.

8.13. Bài tập struct

- ❑ Bài tập 1: Viết chương trình đọc và hiển thị thông tin của sinh viên
 - ❖ Nhập vào từ bàn phím các thông tin của sinh viên như: mã số SV, tên SV, ngành học, điểm trung bình.
 - ❖ Xuất thông tin của sinh viên ra màn hình.

8.13. Bài tập struct

□ Bài tập 2: Sắp xếp danh sách sinh viên theo điểm

- ❖ Sử dụng kết quả của bài 1, sắp xếp sinh viên theo điểm và hiển thị danh sách vừa sắp xếp.

8.13. Bài tập struct

□ Bài tập 3: Tìm kiếm sinh viên trong danh sách

- ❖ Sử dụng kết quả của bài 1, tìm kiếm sinh viên theo mã và hiển thị thông tin sinh viên vừa tìm kiếm.

