



THỰC HỌC – THỰC NGHIỆP



Conceive Design Implement Operate

NHẬP MÔN LẬP TRÌNH

ĐẠI CƯƠNG LẬP TRÌNH

BÀI 8: CẤU TRÚC



- Nắm bắt được kiểu dữ liệu cấu trúc(Structure)
- Khai báo và truy xuất kiểu cấu trúc
- Mảng cấu trúc
- Biết cách sử dụng cấu trúc lồng
- Nắm bắt được kiểu dữ liệu hợp nhất(Union)
- Khai báo và truy xuất kiểu hợp nhất
- So sánh cấu trúc và union



PHẦN 1: STRUCTURE

- ❑ 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) → ??



- ❑ Để lưu được 100 sinh viên, chúng ta cần định nghĩa là 1 kiểu dữ liệu mới, là kiểu dữ liệu Sinh Viên

❖ Ví dụ định nghĩa cấu trúc của sinhvien trong C.

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

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



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

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

- ❖ Ví dụ định nghĩa cấu trúc của sinhvien trong C.

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

❑ Có hai cách để khai báo biến cấu trúc

1. Sử dụng từ khóa struct trong hàm main().

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

2. Khai báo biến cấu trúc tại thời điểm định nghĩa cấu trúc.

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

☐ Có hai cách:

<biên cấu trúc đích> = <biên cấu trúc nguồn>;

<biên 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;
```

- ❑ 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);
```



Gán và truy xuất struct

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

- ❑ Có thể khai báo và sử dụng mảng của structure trong C 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("Nhập thông tin cho 5 sinh viên: \n");  
for (i = 0; i < 5; i++) {  
    printf("Nhập id: ");  
    scanf("%d", &st[i].id);  
    printf("Nhập name: ");  
    scanf("%s", &st[i].name);  
}  
printf("Danh sách sinh viên: \n");  
for (i = 0; i < 5; i++) {  
    printf("Id: %d, Name: %s\n", st[i].id, st[i].name);  
}
```

- ❑ 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.

- ❑ Ví dụ: chúng ta tạo ra 2 cấu trúc và cấu trúc phụ thuộc được sử dụng bên trong cấu trúc chính như một thành viên.

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

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

- ❑ 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;
```

- ❑ 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("Ngay sinh sinh vien (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  
Ngay sinh sinh vien (dd/mm/yyyy): 10/11/1998  
Press any key to continue . . .
```



Cấu trúc lồng nhau

- Nắm bắt được kiểu dữ liệu cấu trúc(Structure)
- Khai báo và truy xuất kiểu cấu trúc
- Mảng cấu trúc
- Cấu trúc lồng





CHÈN QUIZ



PHẦN 2: 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)



❑ Cú pháp:

```
union <tên kiểu 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();
}
```

```
Size of union: 4
date = 24
month = 9
Year = 2014
```

struct

```
struct SinhVien{  
    char ten[100];  
    int tuoi, diem;  
};  
  
int main(){  
    union 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

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.



Sử dụng Union

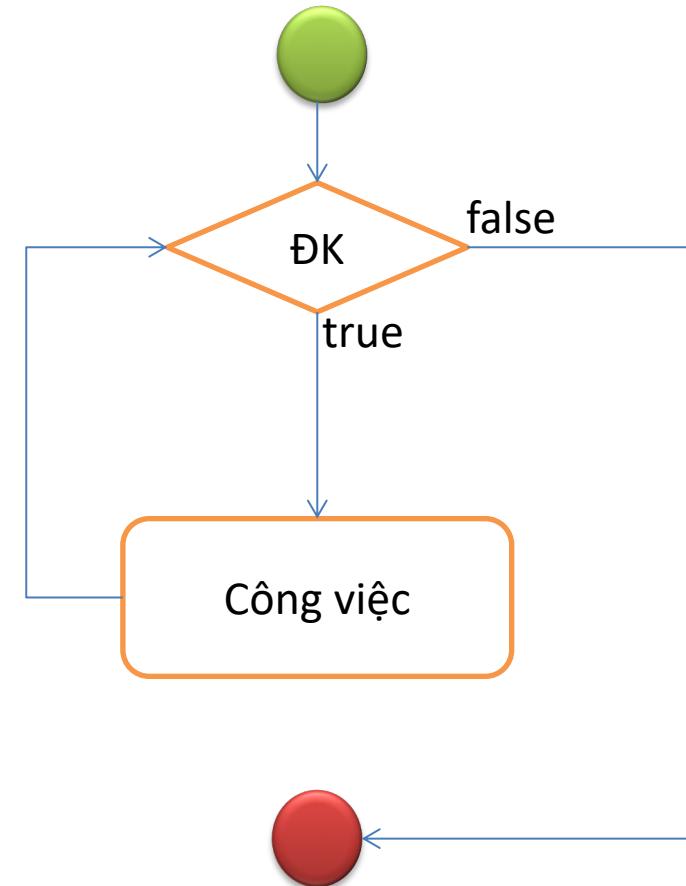
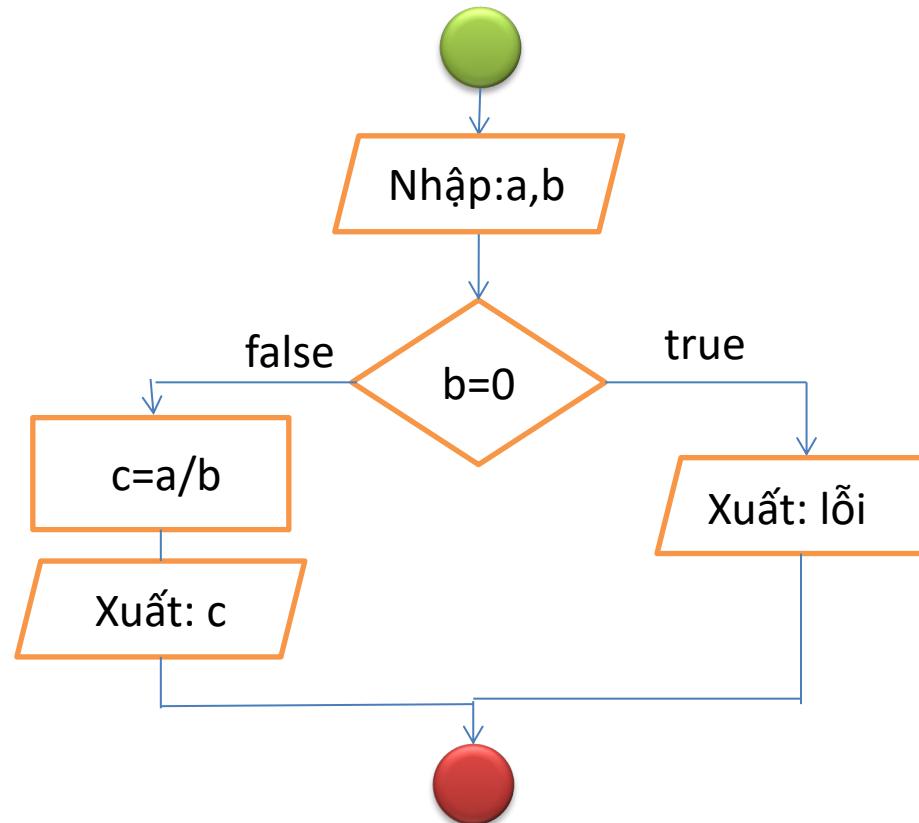
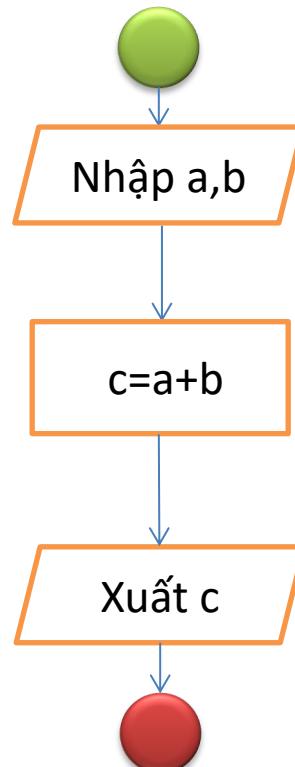


ÔN TẬP KIẾN THỨC MÔN HỌC

- ❑ Kiểu số nguyên
- ❑ Kiểu số thực
- ❑ Kiểu ký tự: 256 ký tự trong bảng mã ASCII.
- ❑ Biến: tên của một vùng nhớ, để chứa dữ liệu.
- ❑ Hằng số: giá trị cố định mà chương trình không thể thay đổi trong quá trình thực thi

CẤU TRÚC CHƯƠNG TRÌNH

- Cấu trúc tuần tự
- Cấu trúc rẽ nhánh
- Cấu trúc lặp



- ❑ Hàm là một đoạn chương trình có tên, đầu vào và đầu ra.
- ❑ Hàm có chức năng giải quyết một số vấn đề chuyên biệt cho chương trình chính.
- ❑ Hàm được gọi nhiều lần với các tham số khác nhau.
- ❑ Hàm được sử dụng khi có nhu cầu:
 - ❖ Tái sử dụng.
 - ❖ Sửa lỗi và cải tiến.

- ❑ Mảng là tập hợp các phần tử cùng kiểu.
- ❑ Mảng có số lượng phần tử cố định và được cấp phát vùng nhớ liên tục.
- ❑ Lợi ích của mảng
 - ❖ Sử dụng mảng để nắm giữ nhiều giá trị thay vì phải khai báo nhiều biến.
 - ❖ Truy xuất nhanh
 - ❖ Dễ dàng đọc dữ liệu từ các phần tử và sắp xếp

- ❑ Chuỗi trong ngôn ngữ lập trình C thực chất là mảng một chiều của các ký tự mà kết thúc bởi một ký tự **null** '\0'.
- ❑ Chuỗi ký tự kết thúc bằng ký tự '\0' (null)
 - ➔ Độ dài chuỗi = kích thước mảng – 1
- ❑ Kiểu char chỉ chứa được một ký tự. Để lưu trữ một chuỗi (nhiều ký tự) ta sử dụng mảng (một chiều) các ký tự.
- ❑ Ví dụ:

```
char hoten[30]; // Dài 29 ký tự
```

- ❑ Cấu trúc(structure), Union trong C là một kiểu dữ liệu người dùng tự định nghĩa cho phép bạn lưu trữ các loại phần tử khác nhau.

- | | |
|---|---|
| <ul style="list-style-type: none">❑ 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. | <ul style="list-style-type: none">❑ 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. |
|---|---|

- Nắm bắt được kiểu dữ liệu hợp nhất(Union)
- Khai báo và truy xuất kiểu hợp nhất
- So sánh cấu trúc và union
- Ôn tập môn học



