Bài 10

Lập trình C
Cấu trúc

Nội dung

- Khái niệm cấu trúc
- Định nghĩa, khai báo và khởi tạo
- Các phép toán với cấu trúc
- Mảng và con trỏ cấu trúc
- Truyền tham số với cấu trúc

Cấu trúc

- **Cấu trúc** là tập hợp các biến **không cùng kiểu** sử dụng **chung một tên**. Mỗi biến được gọi là một **thành viên** hoặc một **trường**.
  - Kiểu dữ liệu do người lập trình định nghĩa.
  - Thường để lưu các mục dữ liệu có liên hệ logic với nhau.
- Ví dụ: thông tin công dân
  - Số căn cước
  - Họ tên
  - Năm sinh
  - …

Định nghĩa cấu trúc

- Lệnh định nghĩa cấu trúc:

struct <tên cấu trúc> {

<kiểu dữ liệu> <tên thành viên 1>;

<kiểu dữ liệu> <tên thành viên 2>;

…

<kiểu dữ liệu> <tên thành viên n>;

};

- ***<tên cấu trúc>*** cũng là một định danh.

```
struct cong_dan {
    char so_cc[13];
    char ho_ten[50];
    int nam_sinh;
};
```

Khai báo biến cấu trúc

- Lệnh khai báo biến cấu trúc:

  struct <tên cấu trúc> <tên biến>;

- ***<tên cấu trúc>*** là một cấu trúc đã được định nghĩa.

```
struct cong_dan c;
```

- Có thể khai báo biến khi định nghĩa cấu trúc:

  struct <tên cấu trúc> {
  …
  } <tên biến>;

```
struct cong_dan {
    char so_cc[13];
    char ho_ten[50];
    int nam_sinh;
} c;
```


Lệnh typedef

- **Lệnh typedef** dùng để đặt tên cho một kiểu dữ liệu

  typedef <kiểu dữ liệu> <tên của kiểu dữ liệu>;

- ***<tên của kiểu dữ liệu>*** cũng là một định danh.

```
typedef int so_nguyen;
typedef int ma_tran_3x3[3][3];
typedef struct cong_dan hoc_sinh;
```

- Sử dụng **tên của kiểu** trong khai báo biến

```
so_nguyen n;     // n là biến so_nguyen
ma_tran_3x3 m;   // m là biến ma_tran_3x3
hoc_sinh h;     // h là biến hoc_sinh
```

Lệnh typedef và cấu trúc

- Định nghĩa cấu trúc với typedef:

typedef struct {

…

} <tên của kiểu cấu trúc>;

hoặc

typedef struct <tên cấu trúc> {

…

} <tên của kiểu cấu trúc>;

```
typedef struct {
    char so_cc[13];
    char ho_ten[50];
    int nam_sinh;
} hoc_sinh;
```

```
typedef struct cong_dan {
    char so_cc[13];
    char ho_ten[50];
    int nam_sinh;
} hoc_sinh;
```


Khởi tạo cho biến cấu trúc

- Có thể khởi tạo giá trị cho các thành viên tại thời điểm khai báo biến cấu trúc.

```
struct cong_dan c = {"109876543210",
                     "Nguyen Van B",
                     1982};
hoc_sinh h = {"012345678910",
              "Tran Van B",
              1980};
```

- Nếu không đủ giá trị thì thành viên bị thiếu giá trị sẽ nhận NULL cho kiểu dạng chuỗi và 0 cho kiểu số.
- Áp dụng qui tắc khởi tạo mảng cho thành viên là mảng.

Các phép toán với cấu trúc (1)

- **Phép toán dấu chấm .** để truy cập thành viên của một biến cấu trúc. Biểu thức truy cập thành viên trực tiếp:

  <tên biến cấu trúc>.<tên thành viên>

- **Phép toán gán =** giữa 2 biến cấu trúc thực hiện sao chép nội dung của một biến cấu trúc vào một biến cấu trúc **cùng loại**.
- **Phép toán ép kiểu ( )** với cấu trúc

  (struct <tên cấu trúc>)

hoặc

(<tên của kiểu cấu trúc>)

Các phép toán với cấu trúc (2)

```
struct cong_dan c = {"109876543210",
                     "Nguyen Van C", 1982};
char ch = c.so_cc[0];
int y = c.nam_sinh;
c.so_cc[0] = '1';
c.nam_sinh = 1982;

struct cong_dan d, e, f;
d = c;
e = (struct cong_dan) {"012345678910",
                       "Tran Van B", y};
f = (hoc_sinh) {"112345678910",
                "Le Van A", y + 2};
```

Mảng và con trỏ cấu trúc

- Khai báo mảng một chiều và con trỏ kiểu cấu trúc

  <kiểu cấu trúc> <tên biến>[kích thước];

  <kiểu cấu trúc> *<tên biến>;

```
struct cong_dan a[2] = {{"012345678910", "Tran Van B"},
                        {"112345678910", "Le Van A"}};
struct cong_dan *p = a;
```

- **Phép toán mũi tên ->** để truy cập thành viên của một biến cấu trúc thông qua con trỏ. Biểu thức truy cập thành viên gián tiếp:

  tên_con_trỏ_cấu_trúc->tên_thành_viên

```
int y = p->nam_sinh; p->nam_sinh = 1982;
```


Truyền tham số với cấu trúc

- Cấu trúc được sử dụng để truyền tham số cho hàm tương tự các kiểu cơ bản.

```
void funcOne(struct cong_dan c) {
    c.nam_sinh = 3000;
}
void funcTwo(struct cong_dan *p) {
    p->nam_sinh = 5000;
}

int main() {
    struct cong_dan a[2] = {{"012345678910", "Tran Van B", 1980},
                            {"112345678910", "Le Van A", 1978}};
    struct cong_dan *p = a;
    funcOne(a[1]);
    funcTwo(p);
    return 0;
}
```