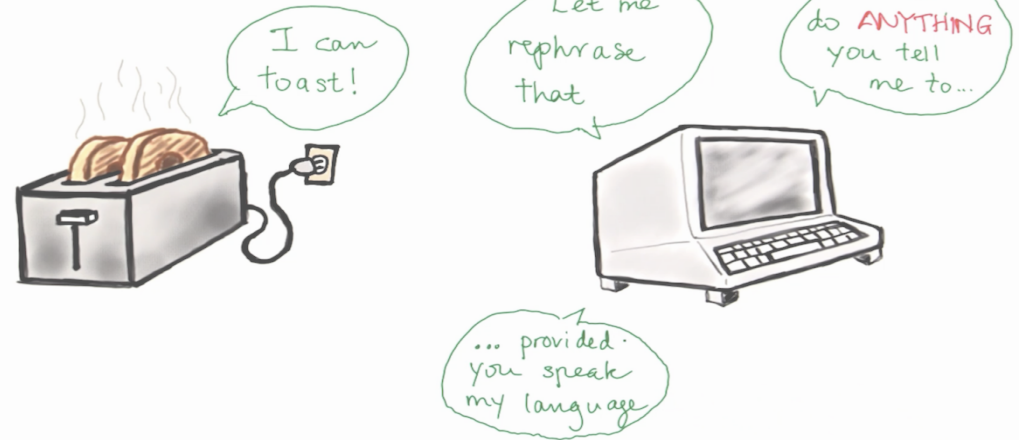


What is programming?



Tuần 10

KIỂU CẤU TRÚC

CT101 – Lập trình căn bản

Khoa CNTT&TT – ĐHCT

Mục đích

- Hiểu được thế nào là kiểu cấu trúc.
- Biết cách khai báo và sử dụng các dạng của kiểu cấu trúc.

Yêu cầu

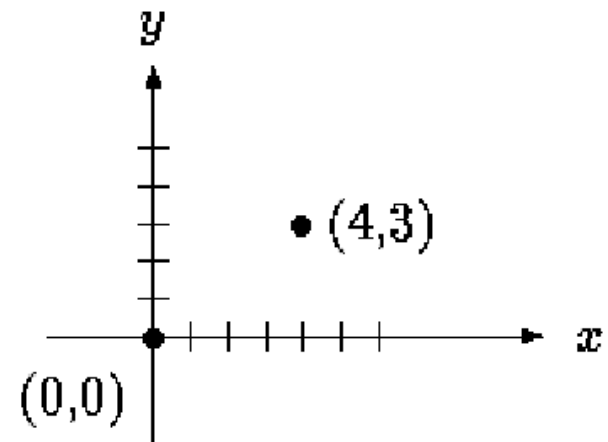
- Hiểu lý thuyết và vận dụng để giải bài tập.
- Hoàn thành hết bài tập.

Nội dung

- Kiểu cấu trúc trong C.
- Các thao tác trên biến kiểu cấu trúc.
- Con trỏ và cấu trúc.

Đặt vấn đề

- **1** nhân viên được mô tả bởi tập các thuộc tính:
 - **id**: kiểu chuỗi
 - **name**: kiểu chuỗi
 - **dob**: kiểu chuỗi
 - **gender**: kiểu ký tự
 - **salary**: kiểu số
- **1** điểm trong không gian 2 chiều:
 - **x**: kiểu số
 - **y**: kiểu số



Cách giải quyết

- Khai báo các biến để lưu trữ 1 nhân viên:

- `char id[8]; // b1400908`
- `char name[50]; // L. T. T. Thao`
- `char dob[10]; // 01/01/1996`
- `char gender; // F`
- `float salary; // 1500`

- Khai báo các biến để lưu trữ 1 điểm:

- `int x; // toạ độ trục hoành`
- `int y; // toạ độ trục tung`

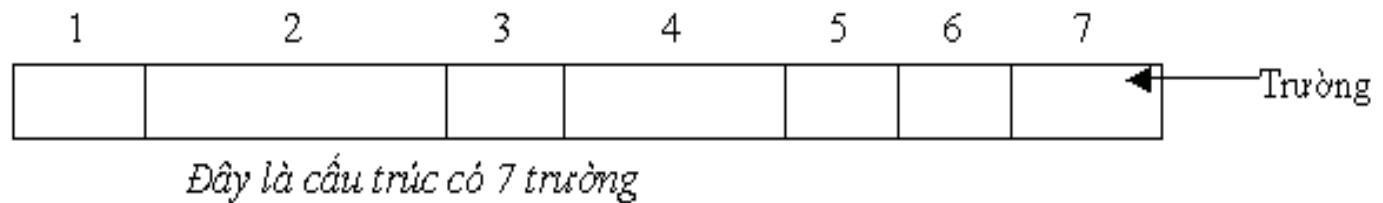
Nhận xét & ý tưởng

- Nhận xét:
 - Khó quản lý khi có nhiều biến, chương trình lớn.
 - Truyền tham số cho hàm quá nhiều
 - Tìm kiếm, sắp xếp, sao chép, ... khó khăn
- Ý tưởng:
 - Gom những thông tin của cùng 1 đối tượng thành một kiểu dữ liệu mới → Kiểu Cấu Trúc (Structure)

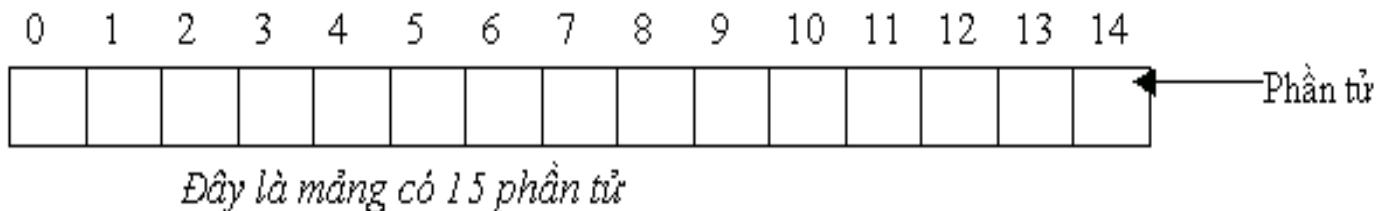
Khái niệm kiểu cấu trúc

- Là kiểu dữ liệu bao gồm nhiều thành phần có kiểu khác nhau, mỗi thành phần được gọi là một trường (field).
- Nó khác với kiểu mảng: các phần tử có cùng kiểu.
- Ví dụ:

1 struct:



1 mảng:



Khai báo (1/4)

- Cú pháp:

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

- Ví dụ:

```
struct Diem2D { int X; int Y; };
```

Khai báo (2/4)

- 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 1>, <tên biến 2>;
```

- Ví dụ:

```
struct Diem2D
{ int X; int Y; } diem2D1, diem2D2;
```

Khai báo (3/4)

- Cú pháp không tường minh:

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

- Khai báo biến:

```
struct <tên kiểu cấu trúc> <tên biến>;
```

- Ví dụ:

```
struct Diem2D { int X; int Y; };  
struct diem2D1, diem2D2;
```

Khai báo (4/4)- 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>;
```

- Khai báo biến:

```
<tên kiểu cấu trúc> <tên biến>;
```

- Ví dụ:

```
typedef struct { int X; int Y; } Diem2D;  
Diem2D diem2D1, diem2D2;
```

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 Diem2D { int X; int Y; }  
diem2D1={0,1}, diem2D2;
```

Gán dữ liệu kiểu cấu trúc

- Có 2 cách
 - $\langle \text{biến cấu trúc đích} \rangle = \langle \text{biến cấu trúc nguồn} \rangle;$
 - $\langle \text{biến cấu trúc đích} \rangle . \langle \text{tên thành phần} \rangle = \langle \text{giá trị} \rangle;$
- Ví dụ:

```
struct Diem2D { int X, int Y; }
```

```
diem2D1={0,1}, diem2D2;
```

```
diem2D2=diem2D1;
```

```
diem2D2.X=diem2D1.X;
```

```
diem2D2.Y=diem2D1.Y;
```

Cấu trúc phức tạp (1/4)

- Thành phần của cấu trúc là cấu trúc khác

```
struct Diem2D { int X, int Y; };
```

```
struct HìnhCN
```

```
{ struct Diem2D TraiTren; struct Diem2D  
PhaiDvoi;
```

```
}hinhchunhat1;
```

- Gán dữ liệu:

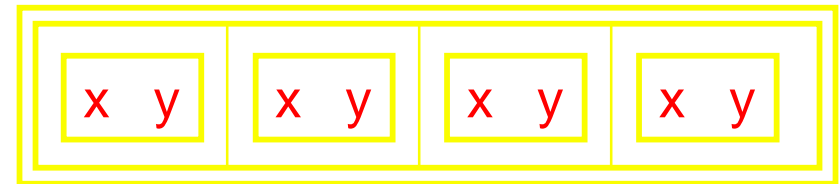
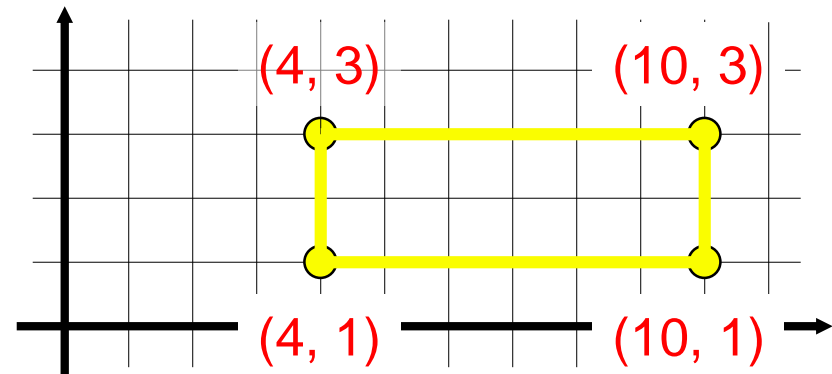
```
hinhchunhat1.TraiTren.X=10;
```

```
hinhchunhat1. PhaiDvoi.X=20;
```

Cấu trúc phức tạp (2/4)

- Thành phần của cấu trúc là mảng

```
struct point{ double x, y;};
struct square{point vertex[4];};
square Sq;
```



Cấu trúc phức tạp (3/4)

- Cấu trúc đệ quy

```
struct Ngươi  
{ char Hoten[30];  
struct Ngươi *NguoiCha, *NguoiMe;  
};
```

Cấu trúc phức tạp (4/4)

- Mảng của cấu trúc

```
struct inventory
{
    int part_no;
    float cost;
    float price;
};
```

```
struct inventory table[4];
```

	part_no	cost	price
table[0]—>	123	10.00	15.00
table[1]—>	.	.	.
table[2]—>	.	.	.
table[3]—>	.	.	.

*(table)	table[0]
(table + 1)	table[1]
(table + 2)	table[2]
(table + 3)	table[3]

Tổng kết

- **Kiểu cấu trúc** là kiểu do người dung định nghĩa.
- **Biến cấu trúc** là biến của kiểu cấu trúc.
- Trong C chuẩn, có thể bỏ từ khoá **struct** khi khai báo biến (hoặc sử dụng **typedef**).
- **Nhập các biến kiểu số thực** trong cấu trúc phải nhập thông qua một biến trung gian.

Kiểm tra kiến thức (1/2)

- 1) Câu nào sau đây dùng để truy cập một biến trong cấu trúc b?
A. b->var;
B. b.var;
C. b-var;
D. b>var;
- 2) Câu nào sau đây dùng để truy cập một biến trong cấu trúc *b?
A. b->var;
B. b.var;
C. b-var;
D. b>var;

Kiểm tra kiến thức (2/2)

- 3) Câu nào sau đây đúng?
- A. `struct {int a;}`
 - B. `struct a_struct {int a;}`
 - C. `struct a_struct int a;`
 - D. `struct a_struct {int a;};`
- 4) Câu nào sau đây dùng để khai báo biến kiểu cấu trúc của cấu trúc foo?
- A. `struct foo;`
 - B. `struct foo var;`
 - C. `foo;`
 - D. `int foo;`

Bài tập tổng kết

- Viết ứng dụng quản lý sinh viên:
 - Khai báo cấu trúc sinh viên gồm thành phần: MSSV, HỌ TÊN, NGÀY SINH, QUÊ QUÁN, ĐIỂM TRUNG BÌNH TÍCH LŨY, ĐIỂM RÈN LUYỆN.
 - Khai báo mảng cấu trúc để quản lý 1 lớp gồm 50 sinh viên.
 - Nhập thông tin của n sinh viên.
 - In thông tin n sinh viên.
 - In thông tin sinh viên thứ n.
 - Tìm thông tin sinh viên theo MSSV.
 - Tìm sinh viên có ĐTBTL/ ĐRL cao nhất.
 - Xếp hạng sinh viên theo ĐTBTL.



CT101 – Lập trình căn bản

Khoa CNTT&TT – ĐHCT