# Chương 8: structures

# Agenda

- Struct
  - Khái niệm
  - Khai báo struct
  - Truy xuất thành phần của struct
  - Gán cấu trúc
  - Mảng cấu trúc
  - Truyền cấu trúc vào hàm
  - Con trỏ cấu trúc
  - typedef

# 1. Struct

1. Khái niệm: **Một struct (1 cấu trúc) là 1 tập hợp các biến, các mảng và được biểu thị bằng 1 tên duy nhất.**
   - Ví dụ: Tập hợp thông tin về 1 sinh viên bao gồm:
     - Tên sinh viên,
     - Năm sinh
     - Địa chỉ thường trú
     - …

# 1.1 Khai báo cấu trúc

- Có 2 kiểu khai báo cấu trúc:
  - **Kiểu 1**:

```
struct structureName
{
    datatype member_1;
    datatype member_2;
    ...
    datatype member_n;
};
```

Struct members

  - structureName: Tên struct
  - datatype: kiểu dữ liệu của biến thành phần

# 1.1 Khai báo cấu trúc

- Khai báo cấu trúc:
  - **Kiểu 1**:
    - Ví dụ 1: Khai báo 1 struct ngày

```
struct ngay
{
      int ngay_thu;
      char  thang[10];
      int nam;
} ;
```

# 1.1 Khai báo cấu trúc

- Khai báo cấu trúc:
  - **Kiểu 1**:
    - Khai báo biến cấu trúc: Sau khi khai báo cấu trúc, ta có 1 kiểu dữ liệu mới (new data type). Muốn có 1 biến có kiểu struct vừa được khai báo, ta phải khai báo biến cấu trúc.
      - **Biến cấu trúc được khai báo giống tất cả những loại biến khác**

```
struct structName variableNames;
```

    - Ví dụ: **struct ngay ngaysinh;**

# 1.1 Khai báo cấu trúc

- Khai báo cấu trúc:
  - **<u>Lưu ý:</u>** *Thành phần bên trong 1 struct có thể là 1 biến có kiểu dữ liệu cơ sở (vd: int, float,..), 1 mảng, hay có thể là 1 biến kiểu struct.*

# 1.1 Khai báo cấu trúc

- Khai báo cấu trúc:
  - **Kiểu 1**:
    - Ví dụ 2: Khai báo struct sinh viên gồm những thành phần sau:
      - Mã sinh viên (10 ký tự)
      - Họ tên (50 ký tự)
      - Ngày sinh (kiểu cấu trúc “ngày”)
      - Quê quán (20 ký tự).
    - Ví dụ 3: Khai báo struct nhân viên gồm những thành phần sau:
      - Họ tên (50 ký tự)
      - Ngày sinh (kiểu cấu trúc “ngày”)

# 1.1 Khai báo cấu trúc

- Khai báo cấu trúc:
  - **Kiểu 1**:
    - Ví dụ 2:
      - Khai báo

```
struct sinhvien
{
      char masv[10];
      char ten[50];
      struct ngay  ngaysinh;
      char quequan[40];
} ;
```

**struct sinhvien svA;**

# 1.1 Khai báo cấu trúc

- Khai báo cấu trúc:
  - **Kiểu 1**:
    - Ví dụ 3:
      - Khai báo

```
struct nhanvien
{
      char ten[50];
      struct ngay  ngaysinh;
      char diachi[50];
      float bacluong;
      struct ngay  ngayvaocty;
} ;
```

```
struct nhanvien nvA;
```

# 1.1 Khai báo cấu trúc

- Khai báo cấu trúc:
  - **Kiểu 2**: Khai báo biến cấu trúc đồng thời với khai báo cấu trúc

```
struct structureName
{
    datatype member_1;
    datatype member_2;
    ...
    datatype member_n;
} variableNames;
```

**Struct members**



- variableNames: tên các biến cấu trúc phân cách

# 1.1 Khai báo cấu trúc

- Khai báo cấu trúc:
  - **Kiểu 2**:
    - Ví dụ 1:
      - Khai báo

```
struct nhanvien
{
      char ten[50];
      struct ngay  ngaysinh;
      char diachi[50];
      float bacluong;
      struct ngay  ngaysinh;
} nvA, nvB;
```

# 1.2 Truy cập các thành phần của biến cấu trúc

- Dạng tổng quát:

**variableName.memberName**

# 1.2 Truy cập các thành phần của biến cấu trúc

- Ví dụ: Xem xét cấu trúc

```
struct coordXY
{
int x;
int y;
} diemA, diemB;
```

  - Để gán tọa độ cho điểm A(100,200), ta dùng các lệnh:
    - diemA.x = 100;
    - diemA.y = 200;

# 1.2 Truy cập các thành phần của biến cấu trúc

- Ví dụ 1: Xem xét cấu trúc

```
struct diem
{
int x;
int y;
} diemA, diemB;
```

  - Để gán tọa độ cho điểm A(100,200), ta dùng các lệnh:
    - diemA.x = 100;
    - diemA.y = 200;

# 1.2 Truy cập các thành phần của biến cấu trúc

- Ví dụ 2: Khai báo 2 điểm A(xA, yA), B (xB, yB),tọa độ của A,B nhập vào từ bàn phím. Tính khoảng cách đoạn thẳng AB biết

$$AB = \sqrt{(x_B - x_A)^2 + (y_B - y_A)^2}$$

# 1.3 Lệnh gán cấu trúc

- Lệnh gán cấu trúc dùng để gán nội dung của 1 biến cấu trúc cho 1 biến cấu trúc khác có cùng kiểu
- Ví dụ: Để gán nội dung của biến cấu trúc điemA cho biến cấu trúc điemB, ta dùng lệnh sau:
  - diemB = diemA

# 1.4 Mảng cấu trúc

- Để khai báo 1 mảng các cấu trúc, đầu giên ta sẽ khai báo cấu trúc trước, sau đó sẽ khai báo 1 mảng của cấu trúc đó.
- Ví dụ: Khai báo mảng point có 100 phần tử:

```
struct diem
{
int x;
int y;
} diemA, diemB;
```

# 1.5 Truyền cấu trúc vào hàm

- Truyền thành viên của cấu trúc vào hàm: Có 2 cách truyền
  - Truyền tham trị: Khi truyền 1 thành phần của 1 cấu trúc vào 1 hàm, thực chất là truyền giá trị của thành phần đó cho tham số hình thức của hàm.
  - Truyền tham chiếu: Để thực hiện việc truyền tham chiếu, ta phải đặt dấu “&” trước tên của thành phần được truyền.

# 1.5 Truyền cấu trúc vào hàm

- Ví dụ 1 – truyền tham trị: Tính khoảng cách đoạn thẳng AB

```
double khoangcach(int xA, int yA, int xB, int yB)
{
	return (sqrt(pow((xB-xA),2)+pow((yB-yA),2)));
}
void main()
{
	struct diem
	{
		int x;
		int y;
	};
	diem A,B; double kcach;
	// Nhập tọa độ 2 điểm A,B
	….
	cout<<"\nKhoang cach giua a diem:”<<khoangcach(A.x, A.y,B.x, B.y);
}
```

# 1.5 Truyền cấu trúc vào hàm

- Ví dụ 2 – truyền tham chiếu: Dịch chuyển điểm A

```
void dichchuyen(int *xA, int *yA, int  delta_x, int delta_y)
{
	*xA = *xA – delta_x;
	*yA = *yA – delta_y;
}
void main()
{
	// Khai báo cấu trúc điểm
	// Nhập tọa độ điểm A
	….
	dichchuyen(&A.x, &A.y, 5, 10);
	cout<<"\nVị trí mới của điểm A: ”;
	cout<<“A.x =”<<A.x;
	cout<<“A.y =”<<A.y;
}
```

# 1.5 Truyền cấu trúc vào hàm

- Truyền toàn bộ biến cấu trúc vào hàm: Có 2 cách truyền
  - Truyền tham trị: Biến cấu trúc được truyền như 1 đối số của hàm.
  - Truyền tham chiếu:

# 1.5 Truyền cấu trúc vào hàm

## Ví dụ 1 – truyền tham trị: Tính khoảng cách đoạn thẳng AB

```
double khoangcach(diem A, diem B)
{
	return (sqrt(pow((B.x-A.x),2)+pow((B.y-A.y),2)));
}
void main()
{
	struct diem
	{
		int x;
		int y;
	};
	diem A,B; double kcach;
	// Nhập tọa độ 2 điểm A,B
	….
	cout<<"\nKhoang cach giua a diem:”<<khoangcach(A,B);
}
```

# 1.5 Truyền cấu trúc vào hàm

- Ví dụ 2 – truyền tham chiếu: Dịch chuyển điểm A.

```
void dichchuyen(diem &A, int  delta_x, int
delta_y)
{
      A.x = A.x- delta_x;
      A.y = A.y – delta_y;
}
void main()
{
      // Khai báo cấu trúc điểm
      // Nhập tọa độ điểm A
      ….
      dichchuyen(A, 5, 10);
      cout<<"\nVị trí mới của điểm A: ";
      cout<<“A.x =”<<A.x;
      cout<<“A.y =”<<A.y;
}
```

# 1.5 Con trỏ đến cấu trúc

- Một con trỏ có thể trỏ đến bất kỳ biến nào, kể cả biến cấu trúc

```
structureName *structurePointers;
```

- Ví dụ:

```
struct diem
{
int x;
int y;
```

# 1.5 Con trỏ đến cấu trúc

- Để tham chiếu đến thành viên của một cấu trúc được trỏ đến bởi 1 con trỏ, ta dùng toán tử "**->**"
- Ví dụ:

```
points *p;

points p = &pointA;

p->x = 100;
```

# 1.7 typedef

- Từ khóa typedef dùng để định nghĩa một kiểu mới dựa trên 1 kiểu dữ liệu có sẵn.
- Dạng tổng quát:

```
typedef  existingType  newType;
```

# 2. enum

- Một **enum là một tập của các tên hằng** nguyên xác định tất cả các giá trị hợp lệ mà một biến của kiểu đó có thể có.
- Cú pháp:

```
enum enumName {enumList} enumVars;
```

  - enum: từ khóa để khai báo enum
  - enumName: Tên của enum
  - enumList: Danh sách các tên hằng nguyên phân cách nhau bởi dấu phẩy
  - enumVars: Tên các biến kiểu enum.

# 2. enum

- enum day {Sun, Mon, Tue, Wed, Thu, Fri, Sat}