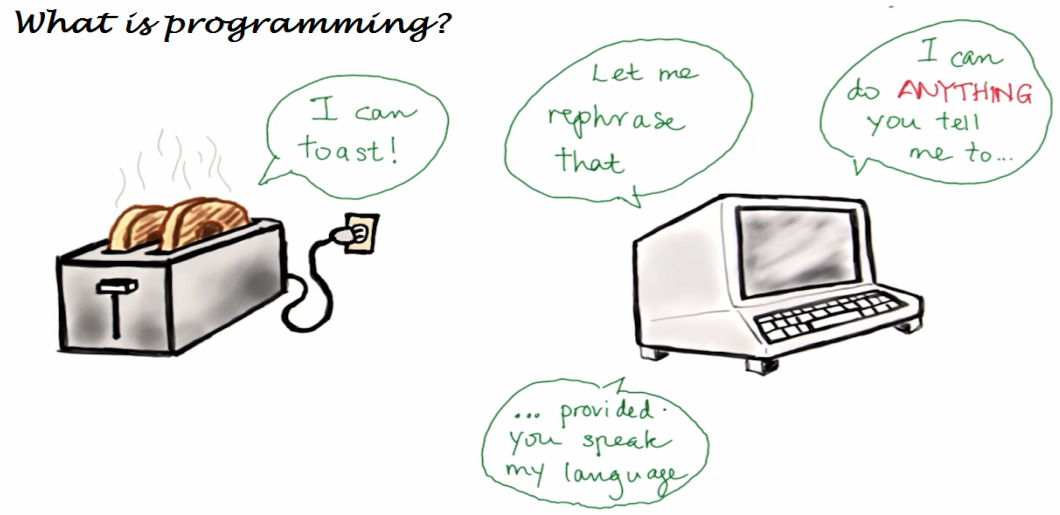


ÔN TẬP KIỂU MẢNG, CẤU TRÚC

Khoa CNTT&TT – ĐHCT

Nội dung

- Chương trình con
- Kiểu mảng (mảng 1 chiều)
- Kiểu cấu trúc
- Kiểu con trỏ
  - Con trỏ đến cấu trúc

Chương trình con

- Sử dụng chương trình con (hàm):
  - Tránh rườm rà và mất thời gian khi viết lặp đi lặp lại nhiều lần những đoạn chương trình giống nhau.
  - Dễ dàng kiểm tra tính đúng đắn của nó trước khi ráp nối vào chương trình chính và do đó việc xác định sai sót để tiến hành hiệu đính trong chương trình chính sẽ thuận lợi hơn.

Chương trình con

- Định nghĩa hàm

```
<kiểu kết quả>  Tên hàm (
                    [<kiểu t số> <tham số>]
                    [,<kiểu t số><tham số>][…])
{
      [Khai báo biến cục bộ]
      [Các câu lệnh thực hiện hàm]
      [return [<Biểu thức>];]
}
```

- Sử dụng hàm/ Gọi hàm

```
<Tên hàm>([Danh sách các tham số])
```

Chương trình con

Các phương pháp truyền tham số

- Truyền giá trị:
  - Là phương pháp truyền tham số mà sau đó hàm được truyền có được một phiên bản được lưu trữ riêng biệt giá trị của các tham số đó.
  - Khi truyền giá trị, thì giá trị gốc (được truyền) sẽ không bị thay đổi cho dù hàm được truyền có thay đổi các giá trị này đi nữa.
- Truyền bằng con trỏ:
  - Là phương pháp truyền tham số mà nó cho phép hàm được truyền tham khảo đến vùng nhớ lưu trữ giá trị gốc của tham số.
  - Nếu ta truyền bằng con trỏ thì giá trị gốc của tham số có thể được thay đổi bởi các mã lệnh bên trong hàm.

Chương trình con

- Xét hàm swap dùng để đổi nội dung 2 biến x, y:

```
#include <stdio.h>
void swap(int x, int y) {
    int temp = x;
    x = y;
    y = temp;
}
int main(){
    int A, B;

    scanf("%d%d", &A, &B);

    printf("A=%d, B=%d\n",A,B);
    swap(A, B);
    printf("A=%d, B=%d\n",A,B);

    return 0;
}
```

Kết quả chạy:

Nhập:

1 3

Trước khi swap:

A=1, B=3

Sau khi swap:

A=1, B=3

Truyền giá trị

Chương trình con

- Xét hàm swap dùng để đổi nội dung 2 biến x, y được truyền bằng con trỏ với dấu * được đặt trước 2 biến x, y:

```
#include <stdio.h>

void swap(int *x, int *y) {
    int temp = *x;
    *x = *y;
    *y = temp;
}

int main()
{
    int A, B;

    scanf("%d%d", &A, &B);

    printf("A=%d, B=%d\n",A,B);
    swap(&A, &B);
    printf("A=%d, B=%d\n",A,B);

    return 0;
}
```

Kết quả chạy:

Nhập:

1 3

Trước khi swap:

A=1, B=3

Sau khi swap:

A=3, B=1

Truyền bằng con trỏ

Mảng 1 chiều

- Mảng đơn hay mảng một chiều là một *mảng tuyến tính*.
- Chỉ có một ***chỉ số*** để biểu diễn vị trí phần tử, mỗi phần tử trong mảng một chiều *không phải là một mảng khác*.

Chỉ số	0	1	2	3	4
Mảng a	**5**	**4**	**33**	**13**	**1**

Mảng 1 chiều

- ***Cú pháp khai báo***:

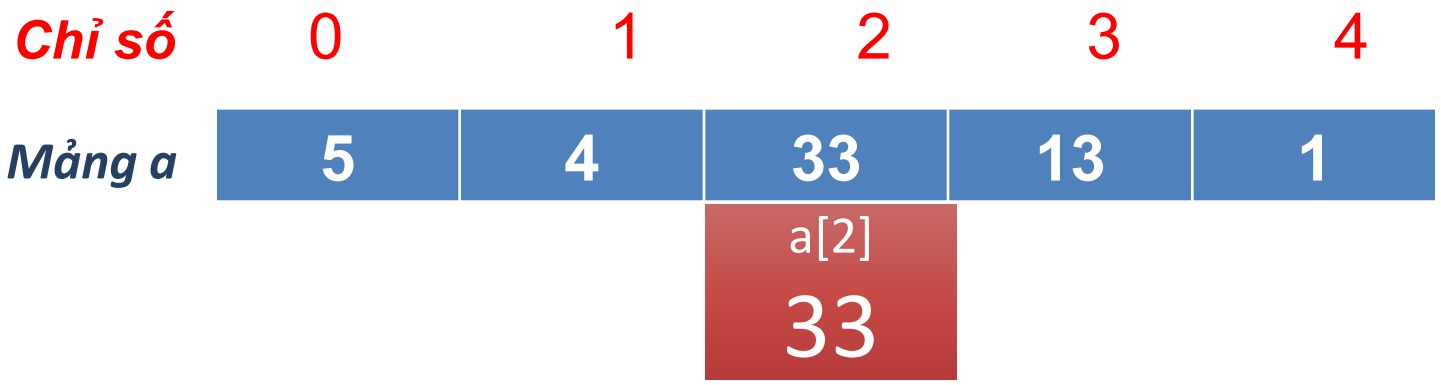
```
<data-type> <array_name> [size];
```

- ***Ví dụ khai báo mảng***:

```
int arr[]; // số phần tử không xác định
int arr[100]; // số phần tử xác định
```

- ***Truy xuất phần tử của mảng:***

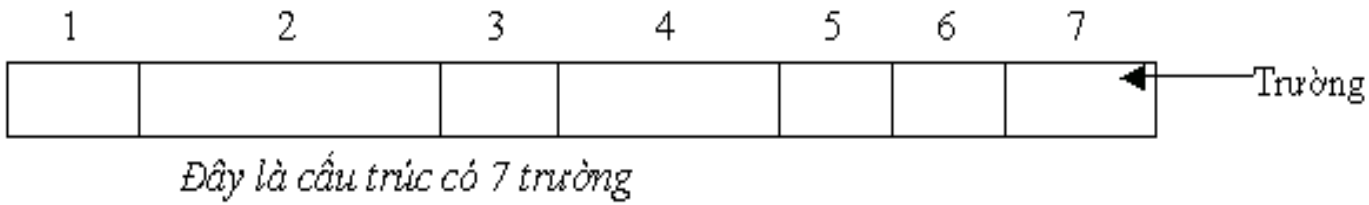
```
<array_name>[index]
```



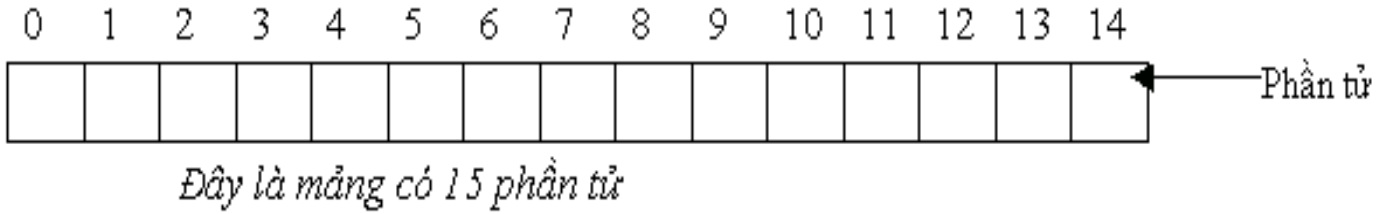
Kiểu cấu trúc

- 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 gồm các phần tử có cùng kiểu.
- Ví dụ:

1 struct:



1 mảng:



Kiểu cấu trúc

- Cú pháp: *Định nghĩa kiểu cấu trúc*

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

- Khai báo biến: *Khai báo biến kiểu cấu trúc*

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

- Ví dụ:

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

Kiểu cấu trúc

- Cú pháp truy xuất từng trường của biến cấu trúc:

```
<tên biến kiểu cấu trúc>.<tên thành phần i>;
```

- Ví dụ:

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

scanf("%d",&diem2D1.X);
scanf("%d",diem2D1.Y);
```

Con trỏ

- Con trỏ là 1 biến đặc biệt: chứa địa chỉ của một ô nhớ (hay 1 biến khác)

  vs. biến thường: chứa dữ liệu.

- Khi con trỏ chứa địa chỉ của 1 ô nhớ (biến), *ta nói là*: con trỏ đang trỏ tới (hay tham chiếu tới) ô nhớ (biến) đó.

- **Tính chất**: con trỏ trỏ tới một ô nhớ (biến)
  ⇒ có thể truy xuất ô nhớ (biến) thông qua con trỏ

Con trỏ

- Cú pháp khai báo:

```
<kiểu dữ liệu> *<tên con trỏ>;
```

  - Kiểu dữ liệu của một con trỏ: xác định *kiểu biến* hay *kiểu của dữ liệu của vùng nhớ* mà con trỏ có thể trỏ tới.
  - Kiểu dữ liệu của con trỏ có thể là bất kỳ kiểu gì.
  - Tên con trỏ: đặt theo qui tắc đặt tên biến.
  - Có thể khai báo nhiều con trỏ trong một câu lệnh.

- Ví dụ:

```
int *p;
float *x, *z;
```

Chú ý: Kích thước của các con trỏ luôn bằng nhau, không phụ thuộc vào kiểu dữ liệu của con trỏ.

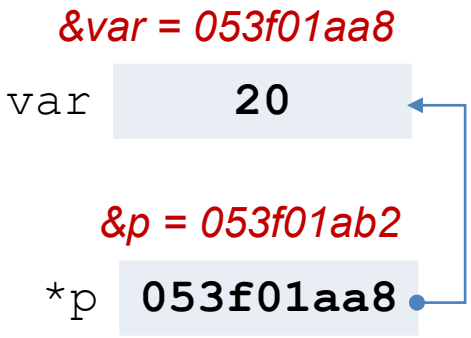
Con trỏ

- Các phép toán trên con trỏ

Phép toán	Ý nghĩa
* *(defererence)*	1. Dùng để khai báo một con trỏ. 2. Truy xuất vùng nhớ (biến) con trỏ đang trỏ đến.
& *(reference)*	Lấy địa chỉ của một biến (địa chỉ vùng nhớ được cấp phát cho biến)

```
int main (){
    int  var = 20;
    int  *p;

    p = &var; //cho p tham chiếu tới var
    printf("%x\n", &var);   ⇒ 53f01aa8
    printf("%x\n", p);      ⇒ 53f01aa8
    printf("%d\n", *p);     ⇒ 20
    printf("%x\n", &p);     ⇒ 53f01ab2
}
```



Con trỏ đến cấu trúc

- Một con trỏ có thể trỏ đến một biến/vùng nhớ có dữ liệu thuộc bất kỳ kiểu gì

  ⇒ một con trỏ có thể trỏ đến một cấu trúc hoặc sử dụng như là một mảng động các phần tử kiểu cấu trúc.

- Ví dụ:

```
typedef struct {
   char hoten[30];
   char ngaysinh[11];
   float diemTBTL;
} Sinhvien;
Sinhvien sv1;
Sinhvien *p = &sv1;
```

Con trỏ đến cấu trúc

- Có 2 cách truy xuất các phần tử của cấu trúc:
  - Dùng toán tử * (dereference) kết hợp với toán tử . (dot)

```
(*p).diemTBTL = 8.5;
strcpy((*p).hoten, "TCAN");
strcpy((*p).ngaysinh, "23/12/78");
```

  - Dùng toán tử -> (arrow operator)

```
p->diemTBTL = 8.5
strcpy(p->hoten, "TCAN");
strcpy(p->ngaysinh, "23/12/78");
```

```
typedef struct {
  char hoten[30];
  char ngaysinh[11];
  float diemTBTL;
} Sinhvien;

Sinhvien sv1;
Sinhvien *p = &sv1;
```