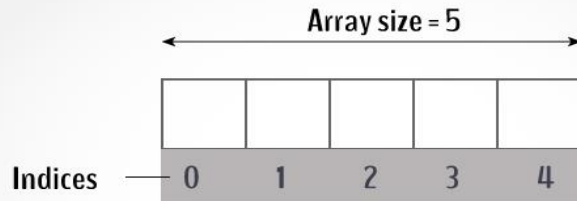


C cơ bản

Mảng



C Arrays



- Giới thiệu mạng
- Xác định một mạng
- Xử lý mạng và khởi tạo mạng trong C
- Mạng hai chiều
- Ưu điểm và nhược điểm của mạng

Phần 1

GIỚI THIỆU Mảng

Ưu điểm của mảng C

- **Tối ưu hóa mã:** Ít mã hơn để truy cập dữ liệu.
- **Dễ dàng đi qua:** Bằng cách sử dụng vòng lặp for, chúng ta có thể truy xuất các phần tử của mảng một cách dễ dàng.
- **Dễ dàng sắp xếp:** Để sắp xếp các phần tử của mảng, chúng ta chỉ cần một vài dòng mã.
- **Truy cập ngẫu nhiên:** Chúng ta có thể truy cập ngẫu nhiên bất kỳ phần tử nào bằng cách sử dụng mảng.

Nhược điểm của mảng C

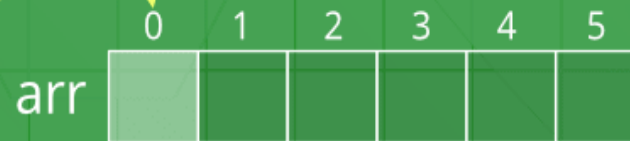
- **Kích thước cố định:** Dù kích thước nào thì chúng ta xác định tại thời điểm khai báo mảng cũng không thể vượt quá giới hạn.

Array in C

array variable

```
arr [ 0 ];
```

index of the element
to be accessed



Phần tử mảng & chỉ mục (2)

- Mỗi thành viên của một **mảng** được xác định bởi chỉ mục hoặc chỉ số duy nhất được gán cho nó
- Kích thước của một **mảng** được xác định bởi số lượng chỉ số cần thiết để xác định duy nhất từng phần tử
- Chỉ số là một số nguyên dương được đặt trong [] ngay sau **mảng** tên
- Một chỉ mục chứa các giá trị số nguyên bắt đầu bằng 0
- Một mảng có 11 phần tử sẽ có dạng:

mảng[0], mảng[1], mảng[2],.... mảng [10]

Phần 2

XÁC ĐỊNH MỘT Mạng

Xác định một mảng (1)

- Một mảng có một số đặc điểm cụ thể và phải được xác định với chúng
- Những đặc điểm này bao gồm -
 - ✓ *Lớp lưu trữ*
 - ✓ *Kiểu dữ liệu của các phần tử trong mảng*
 - ✓ *Tên mảng* (Cho biết vị trí của thành viên đầu tiên của mảng)
 - ✓ *Kích thước mảng* (Một giá trị không đổi)

- Một mảng được định nghĩa giống như cách định nghĩa một biến.
- Thay đổi duy nhất là tên mảng được theo sau bởi một hoặc nhiều biểu thức, được đặt trong dấu ngoặc vuông [], chỉ định kích thước mảng.
- `Storage_Class data_types mảng_name[size]`

`trình phát int[11];`

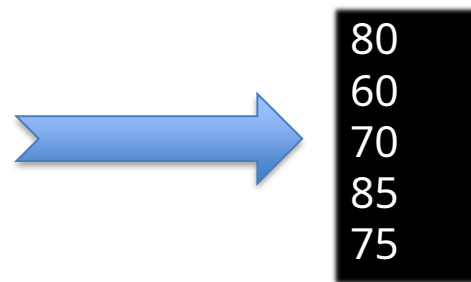
- Tất cả các phần tử của mảng đều có cùng kiểu
- Mỗi phần tử của mảng có thể được sử dụng ở bất cứ nơi nào một biến được cho phép hoặc bắt buộc
- Mỗi phần tử của mảng có thể được tham chiếu bằng cách sử dụng một biến hoặc một biểu thức số nguyên
- Mảng có thể có các kiểu dữ liệu như `int`, `char`, `float` hoặc `double`

Phần 3

XỬ LÝ Mảng VÀ KHỞI TẠO

Cách đơn giản nhất để khởi tạo một **mảng** là bằng cách sử dụng chỉ mục của từng phần tử. Chúng ta có thể khởi tạo từng phần tử của **mảng** bằng cách sử dụng chỉ mục. Hãy xem xét ví dụ sau:

```
# bao gồm <stdio.h>
int chủ yếu(){
    int i = 0;
    int điểm[5];           // khai báo mảng
    điểm[0] = 80;          // khởi tạo mảng
    điểm[1] = 60;
    điểm[2] = 70;
    điểm[3] = 85;
    điểm[4] = 75;
    vì (i = 0; i < 5; i++){ // duyệt mảng
        printf("%d \n", điểm[i]);
    }                       // kết thúc vòng lặp for
    trở lại 0;
}
```



80	60	70	85	75
marks[0]	marks[1]	marks[2]	marks[3]	marks[4]

Initialization of Array

- Chúng ta có thể khởi tạo mảng C tại thời điểm khai báo.
Hãy xem đoạn mã:
- `int mảng[5] = {20, 30, 40, 50, 60};`
- Trong trường hợp như vậy, có **không có yêu cầu xác định kích cỡ**. Vì vậy, nó cũng có thể được viết dưới dạng mã sau:
- `int mảng[] = {20, 30, 40, 50, 60};`

Khởi tạo mảng (2)

```
# bao gồm <stdio.h>
```

```
int chủ yếu(){
```

```
    int tôi = 0;
```

```
    // khai báo và khởi tạo mảng int
```

```
    mảng[5] = { 200, 300, 400, 500,  
600}; // duyệt mảng
```

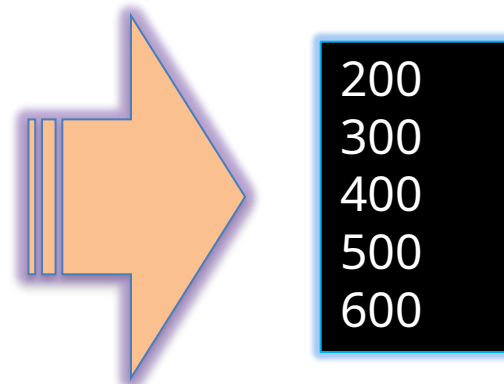
```
    for(i = 0; i < 5; i++){
```

```
        printf("%d \n", mảng[i]);
```

```
    }
```

```
    trở lại 0;
```

```
}
```



- Một **mảng** được xử lý khác với một biến trong C
- Hai **mảng**, ngay cả khi chúng cùng loại và cùng kích thước không thể kiểm tra sự bình đẳng
- Không thể chỉ định một **mảng** trực tiếp đến người khác
- Các giá trị không thể được gán cho một **mảng** về tổng thể, thay vào đó các giá trị được gán cho các phần tử của **mảng**


Xử lý mảng trong C - 2

bao gồm <stdio.h>

trống rỗng chủ yếu()

```
{  
    int mảng[3];  
    int số1 = 2021, số2 = 0; // Nhập đầu  
    vào cho mảng vì(int tôi = 0; tôi < 3;  
    tôi++){  
        printf("\n Nhập giá trị: %d : ", tôi + 1); quét("%d",  
        &arr[i]);  
    }  
    // Gán giá trị cho một phần tử trong mảng mảng[0] =  
    num1;  
    // Lấy giá trị từ một phần tử trong mảng num2 =  
    mảng[1];  
    // Thay đổi giá trị của một phần tử trong mảng mảng [2]  
    = 69;  
    mảng [1] = mảng[2];  
    vì(int tôi = 0; tôi < 3; tôi++){  
        printf("\n Giá trị của phần tử %d là %d", i + 1, mảng[i]);  
    }  
}
```

```
Value of element 1 is 2021  
Value of element 2 is 69  
Value of element 3 is 69  
  
...Program finished with exit code 0  
Press ENTER to exit console.□
```



In ra một mảng từ đầu đến cuối. Ví dụ:

bao gồm <stdio.h>

trống rỗng chủ yếu()

```
{  
    int mảng[5];  
    int số = 2021;  
    // Nhập đầu vào cho mảng vì (int tôi =  
    0; tôi < 5; tôi++){  
        printf("\n Nhập giá trị: %d : ", tôi + 1); quét("%d",  
            &arr[i]);  
    }  
    // In mảng từ đầu đến cuối vì (int tôi = 4;  
    tôi >= 0; Tôi--){  
        printf("\n Giá trị của phần tử %d là %d", i + 1, mảng[i]);  
    }  
}
```

Giá trị đầu vào từ 1 đến 5 =>

```
Enter value: 1 : 1  
  
Enter value: 2 : 2  
  
Enter value: 3 : 3  
  
Enter value: 4 : 4  
  
Enter value: 5 : 5  
  
Value of element 5 is 5  
Value of element 4 is 4  
Value of element 3 is 3  
Value of element 2 is 2  
Value of element 1 is 1  
  
...Program finished with exit code 0  
Press ENTER to exit console.
```

Xử lý mảng trong C - 4

/* Giá trị đầu vào được người dùng chấp nhận vào mảng ary[10]*/

bao gồm <stdio.h>

trống rỗng chủ yếu()

{

int ary[10];

int tôi, tổng cộng, cao; for(i = 0;

i < 10; i++){

printf("\n Nhập giá trị: %d : ", i + 1);

scanf("%d",&ary[i]);

}

/* Hiển thị giá trị cao nhất trong số các giá trị đã nhập

***/** cao = ary[0];

for(i = 1; i < 10; i++){

if(ary[i] > cao)

cao = ary[i];

}

printf("\nGiá trị cao nhất được nhập là %d", high); **/* in giá trị**

trung bình của các giá trị được nhập cho ary[10] */ for(i = 0, total

= 0; i < 10; i++)

tổng = tổng + ary[i];

printf("\nTrung bình cộng của các phần tử của ary là %d",total/i);

}

Xử lý mảng trong C - 5

Hoán đổi các phần tử trong mảng. Ví dụ:

bao gồm <stdio.h>

trống rỗng chủ yếu()

```
{  
    int mảng[5], tạm thời;  
    // Nhập đầu vào cho mảng vì(int tôi =  
    0; tôi < 5; tôi++){  
        printf("\n Nhập giá trị: %d : ", tôi + 1); quét("%d",  
            &arr[i]);  
    }  
    // Hoán đổi phần tử thứ 1 với phần tử thứ 3  
    nhiệt độ = mảng[0];  
    mảng[0] = mảng[2];  
    mảng[2] = nhiệt độ;  
    // In mảng sau khi thay đổi vì(int  
    tôi = 0; tôi < 5; tôi++){  
        printf("\n Giá trị của phần tử %d là %d", i + 1, mảng[i]);  
    }  
}
```



```
Enter value: 1 : 1
```

```
Enter value: 2 : 2
```

```
Enter value: 3 : 3
```

```
Enter value: 4 : 4
```

```
Enter value: 5 : 5
```

```
Value of element 1 is 3
```

```
Value of element 2 is 2
```

```
Value of element 3 is 1
```

```
Value of element 4 is 4
```

```
Value of element 5 is 5
```

```
...Program finished with exit code 0  
Press ENTER to exit console.□
```

Hoán đổi các phần tử giữa hai mảng – mã nguồn:

bao gồm <stdio.h>

trống rỗng chủ yếu(){

int mảng1[4], mảng2[4], tạm thời; //

Nhập đầu vào cho hai mảng vì(int tôi =

0; tôi < 4; tôi++){

printf("\n Nhập giá trị cho 1st-mảng: %d : ", tôi + 1); quét("%d", &arr1[i]);

}

vì(int tôi = 0; tôi < 4; tôi++){

printf("\n Nhập giá trị cho 2thứ-mảng: %d : ", tôi + 1); quét("%d", &arr2[i]);

}

// Hoán đổi hai mảng: vì(int tôi = 0;

tôi < 4; tôi++){

temp = mảng1[i]; mảng1[i] = mảng2[i]; mảng2[i] = tạm thời;

}

// In hai mảng

printf("\n Giá trị của các phần tử trong 1st-mảng sau khi hoán đổi:"); vì(int

tôi = 0; tôi < 4; tôi++){

printf(" %d", mảng1[i]);

printf("\n Giá trị của các phần tử trong 2thứ-mảng sau khi hoán đổi:"); vì(int

tôi = 0; tôi < 4; tôi++){

printf(" %d", mảng2[i]);

}

Xử lý mảng trong C - 7

Hoán đổi phần tử
giữa hai
mảng - đầu ra:



```
Enter value for 1st-array: 1 : 1
Enter value for 1st-array: 2 : 2
Enter value for 1st-array: 3 : 3
Enter value for 1st-array: 4 : 4
Enter value for 2nd-array: 1 : 5
Enter value for 2nd-array: 2 : 6
Enter value for 2nd-array: 3 : 7
Enter value for 2nd-array: 4 : 8

Value of elements in 1st-array after swapping: 5 6 7 8
Value of elements in 2nd-array after swapping: 1 2 3 4

...Program finished with exit code 0
Press ENTER to exit console.
```

Tổng tất cả các phần tử trong mảng. Ví dụ :

bao gồm <stdio.h>

trống rỗng chủ yếu()

```
{
    int mảng[5], tổng = 0;
    // Nhập đầu vào cho mảng vì(int tôi =
    0; tôi < 5; tôi++){
        printf(" Nhập giá trị: %d : ", tôi + 1); quét("%d",
            &arr[i]);
    }
    // Thêm tất cả các phần tử vì(int
    tôi = 0; tôi < 5; tôi++){
        tổng = tổng + mảng[i];
    }
    printf("\n Tổng các phần tử là: %d", Tổng);
}
```



```
Enter value: 1 : 1
Enter value: 2 : 2
Enter value: 3 : 3
Enter value: 4 : 4
Enter value: 5 : 5

Sum of elements is: 15

...Program finished with exit code 0
Press ENTER to exit console.
```

phần 4

Mảng HAI CHIỀU

- Multi- đơn giản nhất và được sử dụng phổ biến nhất mảng chiều là mảng hai chiều.
- Mảng hai chiều có thể được coi là một mảng gồm hai mảng một chiều.
- Mảng hai chiều trông giống như bảng thời gian của đường sắt gồm có hàng và cột.

- Cú pháp khai báo mảng 2D như sau:

`data_type array_name[hàng][cột];`

- Ví dụ:

`int mảng[3][4];`

```
int ary[3][4] =  
{1,2,3,4,5,6,7,8,9,10,11,12};
```

Kết quả của phép gán trên sẽ như sau:

<code>ary [0] [0] = 1</code>	<code>ary [0] [1] = 2</code>	<code>ary [0] [2] = 3</code>	<code>ary [0] [3] = 4</code>
<code>ary [1] [0] = 5</code>	<code>ary [1] [1] = 6</code>	<code>ary [1] [2] = 7</code>	<code>ary [1] [3] = 8</code>
<code>ary [2] [0] = 9</code>	<code>ary [2] [1] = 10</code>	<code>ary [2] [2] = 11</code>	<code>ary [2] [3] = 12</code>

Một kiểu khởi tạo khác:

```
intmảng [3][4]= {  
  
    {1,2,3,4},  
    {5,6,7,8},  
    {9,10,11,12}  
};
```

Kết quả của bài làm sẽ như sau:

<code>ary[0][0] =1</code>	<code>ary[0][1]=2</code>	<code>ary[0][2]=3</code>	<code>ary[0][3]=0</code>
<code>ary[1][0]=4</code>	<code>ary[1][1]=5</code>	<code>ary[1][2]=6</code>	<code>ary[1][3]=0</code>
<code>ary[2][0]=7</code>	<code>ary[2][1]=8</code>	<code>ary[2][2]=3</code>	<code>ary[2][3]=0</code>

Mảng chuỗi hai chiều được khai báo như sau:

`char str_ary[25][30];`

Mảng hai chiều – Ví dụ 1

```
# bao gồm<stdio.h>
int chủ yếu(){
    int tôi=0,j=0;
    int mảng[4][3]={1,2,3},{2,3,4},{3,4,5},{4,5,6}}; //duyet mảng
    2D
    for(i=0; i<4; i++){
        for(j=0; j<3; j++){
            printf("arr[%d] [%d] = %d \n",i,j,arr[i][j]); }//cuối j

    // kết thúc của tôi
    trở lại0;
}
```

In ra tất cả các phần tử của
mảng hai chiều

mảng[0][0] = 1
mảng[0][1] = 2
mảng[0][2] = 3
mảng[1][0] = 2
mảng[1][1] = 3
mảng[1][2] = 4
mảng[2][0] = 3
mảng[2][1] = 4
mảng[2][2] = 5
mảng[3][0] = 4
mảng[3][1] = 5
mảng[3][2] = 6

Mảng hai chiều – Ví dụ 2

Lưu trữ các phần tử trong ma trận và in nó:

```
# bao gồm<stdio.h>
int chủ yếu(){
    int mảng[3][3],i,j;
    cho (i=0; i<3; i++) {
        cho (j=0; j<3; j++) {
            printf("Nhập a[%d][%d]: ", tôi, j); quét("%d"
                , &arr[i][j]);
        }
    }
    printf("\n in các phần tử ....\n"); cho (i=0; i<3; i++) {

        printf("\n");
        cho (j=0; j<3; j++) {
            printf("%d\t", mảng[i][j]);
        }
    }
    trở lại0;
}
```

```
Enter a[0][0]: 1
Enter a[0][1]: 2
Enter a[0][2]: 3
Enter a[1][0]: 4
Enter a[1][1]: 5
Enter a[1][2]: 6
Enter a[2][0]: 7
Enter a[2][1]: 8
Enter a[2][2]: 9
```

printing the elements

1	2	3
4	5	6
7	8	9

...Program finished with exit code 0
Press ENTER to exit console.

Cảm ơn

Hỏi đáp

