

Cấu trúc dữ liệu và giải thuật

TS. Phạm Tuấn Minh

Khoa Công nghệ Thông tin, Đại học Phenikaa

minh.phamtuan@phenikaa-uni.edu.vn

<https://sites.google.com/site/phamtuanminh/>

Chương 2: Mảng và danh sách liên kết

❑ Cấu trúc lưu trữ mảng

- Mảng một chiều
- Mảng nhiều chiều
 - Khai báo mảng, khởi tạo thao tác trên mảng nhiều chiều
 - Mảng nhiều chiều là tham số của hàm
 - Sử dụng mảng một chiều trong mảng hai chiều
 - Toán tử sizeof

❑ Danh sách liên kết

- ❑ Ngăn xếp
- ❑ Hàng đợi

Khai báo mảng nhiều chiều

- Ví dụ mảng 2 chiều:

`int x[3][5];` // mảng 3 phần tử của các mảng 5 phần tử

- Ví dụ mảng 3 chiều

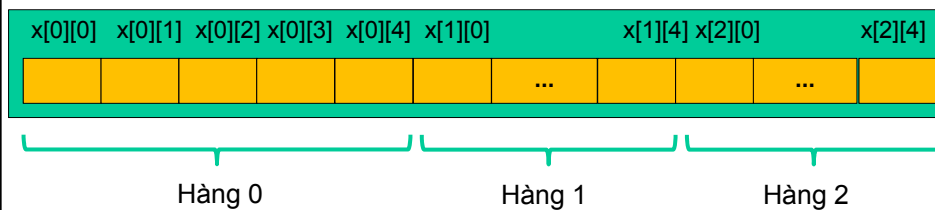
`char x[3][4][5];` // mảng 3 phần tử của các mảng 4 phần tử của các mảng 5 phần tử

1-3

Mảng nhiều chiều

	Cột 0	Cột 1	Cột 2	Cột 3	Cột 4
Hàng 0	<code>x[0][0]</code>	<code>x[0][1]</code>	<code>x[0][2]</code>	<code>x[0][3]</code>	<code>x[0][4]</code>
Hàng 1	<code>x[1][0]</code>	<code>x[1][1]</code>	<code>x[1][2]</code>	<code>x[1][3]</code>	<code>x[1][4]</code>
Hàng 2	<code>x[2][0]</code>	<code>x[2][1]</code>	<code>x[2][2]</code>	<code>x[2][3]</code>	<code>x[2][4]</code>

Tổ chức lưu trữ trong Bộ nhớ



1-4

Khởi tạo mảng nhiều chiều

- Khởi tạo mảng nhiều chiều:

```
int x[2][2] = { {1, 2}, // hàng thứ nhất  
               {6, 7} }; // hàng thứ hai
```

hoặc

```
int x[2][2] = {1, 2, 6, 7};
```

- Khởi tạo một phần

```
int exam[3][3] = {{1, 2, {4}}, {5,7}};
```

```
int exam[3][3] = {1, 2, 4, 5,7};
```

//tương đương

```
int exam[3][3] = {{1, 2, 4}, {5,7}};
```

1-5

Khởi tạo mảng nhiều chiều

- Có thể bỏ qua **chiều ngoài cùng** vì trình biên dịch có thể nhận biết, ví dụ

```
int arr[ ][3][2] = {  
    { {1,1}, {0,0}, {1,1} },  
    { {0,0}, {1,2}, {0,1} }  
};
```

tạo ra mảng có chiều là [2][3][2]

- Khai báo sau là không hợp lệ

```
int wrong_arr[ ][ ] = {1,2,3,4};
```

1-6

Thao tác trên mảng nhiều chiều - Dùng chỉ số mảng

```
#include <stdio.h>
int main()
{ // declare an array with initialization
    int array[3][3]={
        {5, 10, 15},
        {10, 20, 30},
        {20, 40, 60}
    };
    int row, column, sum;
    /* compute sum of row - traverse each row first */
    for (row = 0; row < 3; row++) { // nested loop
        /* for each row - compute the sum */
        sum = 0;
        for (column = 0; column < 3; column++)
            sum += array[row][column]; // using array indexes
        printf("The sum of elements in row %d is %d\n", row+1, sum);
    }
}
```

Thao tác trên mảng nhiều chiều - Dùng chỉ số mảng

```
/* compute sum of each column */
for (column = 0; column < 3; column++) {
    sum = 0;
    for (row = 0; row < 3; row++)
        sum += array[row][column];
    printf("The sum of elements in column %d is %d\n", column+1, sum);
}
return 0;
}
```

Output

The sum of elements in row 1 is 30
The sum of elements in row 2 is 60
The sum of elements in row 3 is 120
The sum of elements in column 1 is 35
The sum of elements in column 2 is 70
The sum of elements in column 3 is 105

Chương 2: Mảng và danh sách liên kết

- ❑ Cấu trúc lưu trữ mảng
 - Mảng một chiều
 - Mảng nhiều chiều
 - Khai báo mảng, khởi tạo thao tác trên mảng nhiều chiều
 - **Mảng nhiều chiều là tham số của hàm**
 - Sử dụng mảng một chiều trong mảng hai chiều
 - Toán tử sizeof
- ❑ Danh sách liên kết
- ❑ Ngăn xếp
- ❑ Hàng đợi

1-9

Mảng nhiều chiều là tham số của hàm

- ❑ Định nghĩa của hàm có mảng 2 chiều là tham số như sau:

```
void fn(int ar2[2][4])  
{  
    ...  
}
```

```
void fn(int ar2[][4])  
{  
    ...  
}
```

- ❑ Trong định nghĩa trên, chiều thứ nhất có thể bỏ qua vì trình biên dịch cần thông tin của mọi chiều trừ chiều thứ nhất của mảng

1-10

Tại sao chiều thứ nhất có thể bỏ qua

- Ví dụ, lệnh gán

ar2[1][3] = 100;

yêu cầu trình biên dịch địa chỉ của **ar2[1][3]** và ghi giá trị 100 vào địa chỉ này.

Để tính địa chỉ, thông tin về chiều phải chuyển cho trình biên dịch.

- Giả sử định nghĩa ar2 như sau

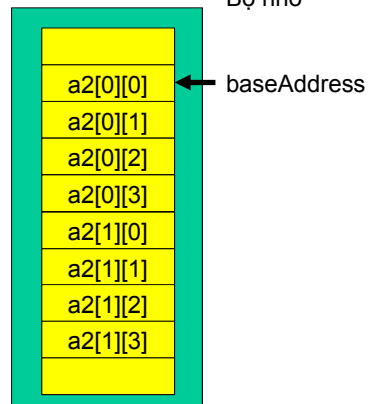
int **ar2[D1][D2];**

Địa chỉ của ar2[1][3] được tính như sau

baseAddress + row * **D2** + column

==> baseAddress + 1 * 4 + 3

==> baseAddress + 7



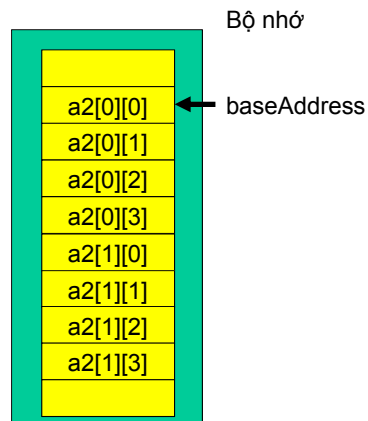
Tại sao chiều thứ nhất có thể bỏ qua

- baseAddress là địa chỉ trỏ tới phần tử đầu tiên của ar2
- Vì không cần D1 khi tính địa chỉ, nên có thể bỏ qua giá trị của chiều đầu tiên khi định nghĩa hàm có mảng là tham số
- Prototype của hàm như sau

void fn(int **ar2[2][4];**

hoặc

void fn(int **ar2[][4];**



Truyền mảng 2 chiều là tham số hàm

```
#include <stdio.h>
int sum_rows(int ar[ ][3]);
int sum_columns(int ar[ ][3]);
int main()
{
    int array[3][3]= {
        {5, 10, 15},
        {10, 20, 30},
        {20, 40, 60}
    };
    int total_row, total_column;
    total_row = sum_rows(array); // sum of all rows
    total_column = sum_columns(array); // all columns
    printf("The sum of all elements in rows is %d\n", total_row);
    printf("The sum of all elements in columns is %d\n", total_column);
    return 0;
}
```

Output

The sum of all elements in rows is 210
The sum of all elements in columns is 210

Truyền mảng 2 chiều là tham số hàm

```
int sum_rows(int ar[ ][3])
{
    int row, column;
    int sum=0;
    for (row = 0; row < 3; row++){
        for (column = 0; column < 3; column++){
            sum += ar[row][column];
        }
    }
    return sum;
}
int sum_columns(int ar[ ][3])
{
    int row, column;
    int sum=0;
    for (column = 0; column < 3; column++){
        for (row = 0; row < 3; row++){
            sum += ar[row][column];
        }
    }
    return sum;
}
```

Bỏ qua khai báo
chiều thứ nhất

Chương 2: Mảng và danh sách liên kết

❑ Cấu trúc lưu trữ mảng

- Mảng một chiều
- Mảng nhiều chiều
 - Khai báo mảng, khởi tạo thao tác trên mảng nhiều chiều
 - Mảng nhiều chiều là tham số của hàm
 - **Sử dụng mảng một chiều trong mảng hai chiều**
 - Toán tử Sizeof

❑ Danh sách liên kết

❑ Ngăn xếp

❑ Hàng đợi

1-15

Sử dụng mảng 1 chiều trong mảng 2 chiều

```
#include <stdio.h>
void display1(int *ptr, int size);
void display2(int ar[ ], int size);

int main()
{
    int array[2][4] = { 0, 1, 2, 3, 4, 5, 6, 7 };
    int i;

    for (i=0; i<2; i++) { /* as 2-D Array */
        display1(array[i], 4);
        display2(array[i], 4);
    }

    display1(array, 8); /* as 1-D array */
    display2(array, 8); /* as 1-D array */
    return 0;
}
```

Output:

Display1 result: 0 1 2 3

Display2 result: 0 5 10 15

Display1 result: 4 5 6 7

Display2 result: 20 25 30 35

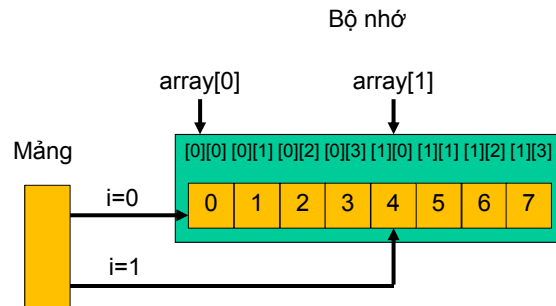
Display1 result: 0 1 2 3 4 5 6 7

Display2 result: 0 5 10 15 20 25 30 35

Sử dụng mảng 1 chiều trong mảng 2 chiều

```
void display1(int *ptr, int size)
{
    int j;
    printf("Display1 result: ");
    for (j=0; j<size; j++)
        printf("%d ", *ptr++);
    putchar('\n');
}
```

```
void display2(int ar[ ], int size)
{
    int k;
    printf("Display2 result: ");
    for (k=0; k<size; k++)
        printf("%d ", 5*ar[k]);
    putchar('\n');
}
```



Chương 2: Mảng và danh sách liên kết

□ Cấu trúc lưu trữ mảng

- Mảng một chiều
- Mảng nhiều chiều
 - Khai báo mảng, khởi tạo thao tác trên mảng nhiều chiều
 - Mảng nhiều chiều là tham số của hàm
 - Sử dụng mảng một chiều trong mảng hai chiều
 - **Toán tử Sizeof**

□ Danh sách liên kết

- Ngăn xếp
- Hàng đợi

Toán tử Sizeof

- ❑ sizeof() là toán tử trả về **kích thước** (theo byte) của toán hạng. Cú pháp
sizeof(operand)
hoặc
sizeof operand
- ❑ **operand** có thể là
 - int, float,... tên kiểu dữ liệu phức tạp, tên biến, tên mảng

1-19

Toán tử sizeof

```
#include <stdio.h>
int sum(int a[], int);
int main(){
    int ar[6] = {1,2,3,4,5,6};
    int total;
    printf("Array size is %d\n", sizeof(ar)/sizeof(ar[0]));
    total = sum(ar, 6);
    return 0;
}
int sum ( int a[], int n ) {
    int i, total=0;
    printf("Size of a = %d\n", sizeof(a));
    for ( i=0; i<n ; i++)
        total += a[i];
    return total;
}
```

Output:

Array size is 6 (i.e. 24/4=6)
Size of a = 4

sizeof cho **biến con trỏ** (i.e., a) cho kết quả là kích thước của con trỏ

Tóm tắt: Mảng 2 chiều

	Cột 0	Cột 1	Cột 2	Cột 3	Cột 4	
Hàng 0	x[0][0]	x[0][1]	x[0][2]	x[0][3]	x[0][4]	Vị trí bộ nhớ liên tục
Hàng 1	x[1][0]	x[1][1]	x[1][2]	x[1][3]	x[1][4]	
Hàng 2	x[2][0]	x[2][1]	x[2][2]	x[2][3]	x[2][4]	

```
// Printing array elements
#include <stdio.h>
int main ( ) {
    int ar[3][3]= {
        {5, 10, 15},
        {10, 20, 30},
        {20, 40, 60}
    };
    int i, j;
    /* using index – nested loop*/
    printf("\n");
    for (i=0; i<3; i++)
        for (j=0; j<3; j++)
            printf("%d ", ar[i][j]);
    printf("\n");
    return 0;
}
```

[Dùng chỉ số](#)

```
// Print array elements
#include <stdio.h>
#define SIZE 9
int main ( ) {
    int ar[3][3]= {
        {5, 10, 15},
        {10, 20, 30},
        {20, 40, 60}
    };
    int i, *ptr;
    ptr = ar;
    /* using pointer – looping */
    for (i=0; i<SIZE; i++)
        printf("%d ", *ptr++);
    printf("\n");
    return 0;
}
```

[Dùng con trỏ như mảng 1 chiều](#)

Chương 2: Mảng và danh sách liên kết

□ Cấu trúc lưu trữ mảng

- Mảng một chiều
- Mảng nhiều chiều
 - Khai báo mảng, khởi tạo thao tác trên mảng nhiều chiều
 - Mảng nhiều chiều là tham số của hàm
 - Sử dụng mảng một chiều trong mảng hai chiều
 - Toán tử sizeof

□ Danh sách liên kết

- Ngăn xếp
- Hàng đợi

Cấu trúc dữ liệu và giải thuật

- Nội dung bài giảng được biên soạn bởi TS. Phạm Tuấn Minh.

1-23