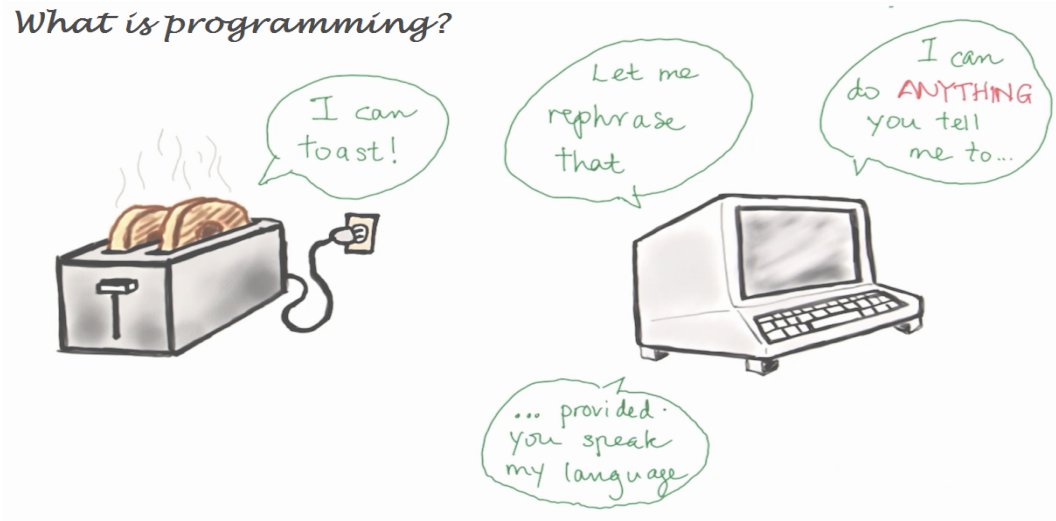


Tuần 6

MẢNG

CT101 – Lập trình căn bản

Khoa CNTT&TT – ĐHCT

Mục đích

- Cung cấp các khái niệm về mảng, cấu trúc mảng
- Cung cấp khái niệm về mảng một chiều và nhiều chiều
- Giới thiệu cách khai báo và khởi tạo mảng
- Giới thiệu cách nhập và xuất mảng

Yêu cầu

- Nắm vững các khái niệm về mảng, cấu trúc mảng
- Hiểu các khái niệm về mảng một chiều và nhiều chiều
- Biết cách khai báo và khởi tạo mảng
- Biết cách nhập và xuất mảng, cách sử dụng mảng trong chương trình cơ bản
- Biết cách vận dụng mảng để giải quyết các bài toán có sử dụng dữ liệu mảng
- Biết vận dụng các thuật toán cơ bản để thao tác trên mảng

Nội dung

- Mảng
- Mảng một chiều
- Mảng nhiều chiều

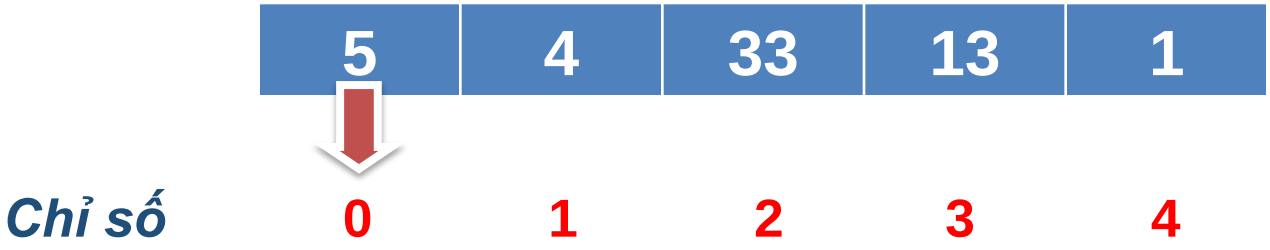
Tại sao ta cần kiểu mảng?

- Trong lập trình đôi khi chúng ta phải đối mặt với việc phải lưu trữ các biến có cùng kiểu dữ liệu như: ***num0***, ***num1***,… và ***num99***
- Mảng giúp nhóm các biến dữ liệu trên thành một biến duy nhất và sẽ được truy cập bằng chỉ số **nums[0]**, ***nums[1]***, … ***nums[99]***.
- Sử dụng mảng giúp người lập trình dễ dàng quản lý dữ liệu hơn thay cho việc khai báo từng giá trị riêng lẻ.

Khái niệm về Mảng

- Mảng là tập hợp các phần tử có cùng một kiểu dữ liệu.
- Các phần tử trong mảng được lưu trữ nối tiếp nhau trong bộ nhớ
- Các phần tử trong mảng có thể được truy cập thông qua các chỉ số

Các phần tử trong mảng

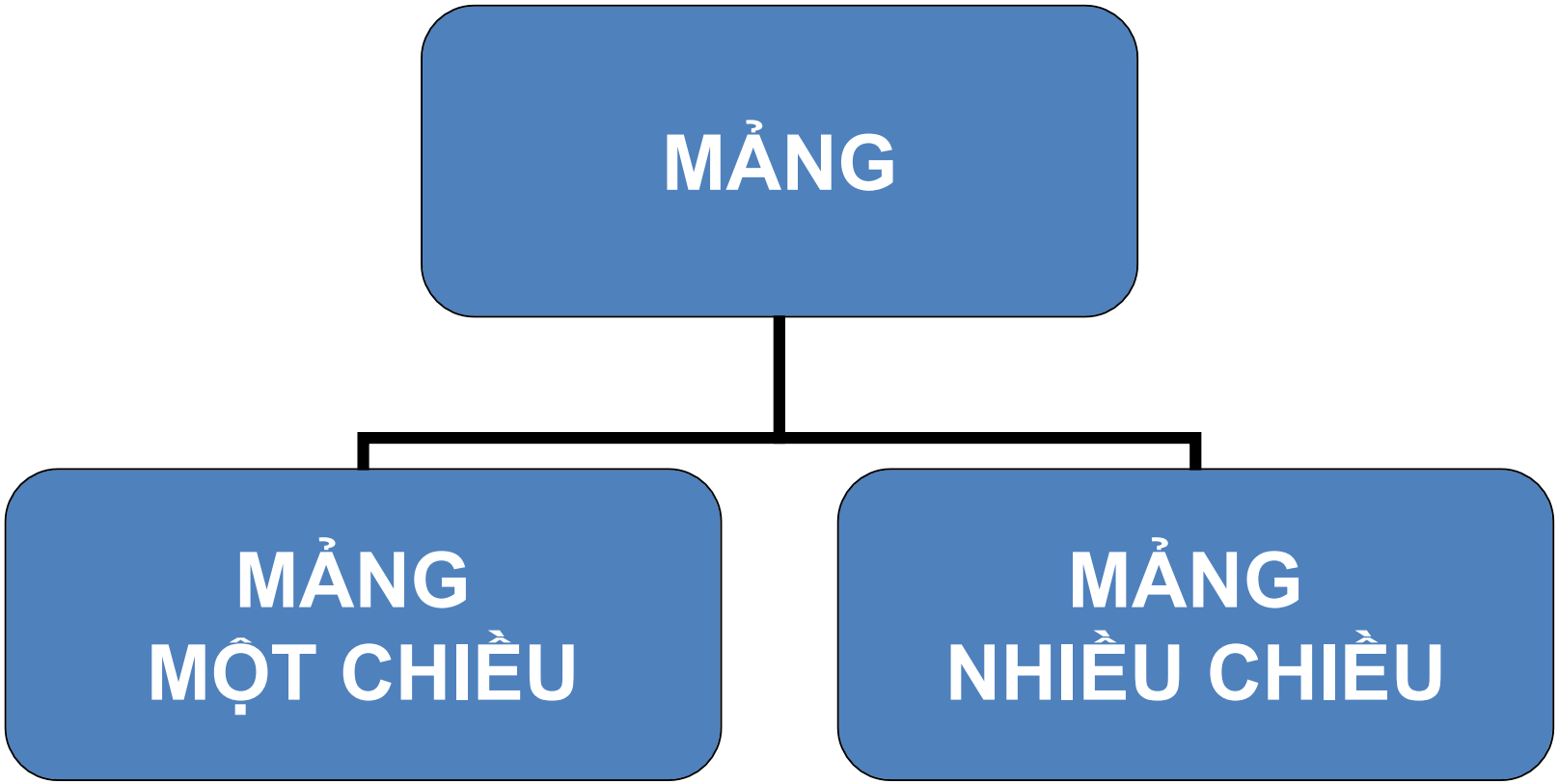



```
a[0] = 4;
a[1] = 5;
a[2] = 33;
a[3] = 13;
a[4] = 1;
```

4, 5, 33, 13, 1 là các giá trị dữ liệu thực sự trong mảng.

0, 1, 2, 3, 4 là các chỉ số để chỉ thứ tự của các phần tử

Khái niệm về Mảng



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*.

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
```

- ***Vừa khai báo vừa gán giá trị:***

```
int iarr[3] = {2, 3, 4} ;
char carr[20] = “c4learn” ;
float farr[3] = {12.5,13.5,14.5} ;
```

- ***Xác định số phần tử của mảng:***

```
size = sizeof(anArray)/sizeof(<kiểu phần tử>)
```

Mảng 1 chiều

- Ta thường dùng vòng lặp ***for*** để thao tác các phần tử trong mảng 1 chiều

```c
#include <stdio.h>

int main () {
    int n[ 10 ]; /* Khai báo mảng n 10 số nguyên */
    int i,j;

    /* Khởi tạo các phần tử mảng n */
    for ( i = 0; i < 10; i++ ) {
        n[ i ] = i + 100; /* Đặt giá trị phần tử là chỉ số + 100 */
    }

    /* Xuất các phần tử của mảng */
    for (j = 0; j < 10; j++ ) {
        printf("Element[%d] = %d\n", j, n[j] );
    }
    return 0;
}
```

Mảng nhiều chiều

- Mảng nhiều chiều có nhiều hơn một chỉ số để biểu diễn một phần tử trong mảng.
- Mảng nhiều chiều cũng được gọi là **ma trận**

Ví dụ mảng 2 chiều có kích thước 6x6

Trong bộ nhớ, các phần tử mảng nhiều chiều được sắp xếp liên tiếp nhau thành 1 hàng.

Trong ví dụ bên, bộ nhớ sẽ lưu trữ từ 1 đến 36 thành một dãy liên tiếp.

	0	1	2	3	4	5
0	1	2	3	4	5	6
1	7	8	9	10	11	12
2	13	14	15	16	17	18
3	19	20	21	22	23	24
4	25	26	27	28	29	30
5	31	32	33	34	35	36

a[1][2]

9

Mảng nhiều chiều

- ***Cú pháp:***

```
<data_type> <array_name> [size1][size2].....[sizen];
```

- ***Khai báo mảng 2 chiều***

```
int a[3][4]; // Ma trận gồm 3 dòng x 4 cột => 12 phần tử
```

- ***Khai báo số lượng phần tử không xác định:***

```
int a[][][5]; // Ta vẫn phải khai báo chiều cuối cùng
```

- ***Vừa khai báo vừa gán giá trị:***

```
int a[3][2] = {
                  { 1 , 4 },
                  { 5 , 2 },
                  { 6 , 5 }
              };
```

Mảng nhiều chiều

- Ta sử dụng các vòng lặp ***for lồng nhau*** để lướt qua các phần tử của mảng nhiều chiều

Ví dụ: Với mảng 2 chiều

```
#include<stdio.h>
int main() {
     int i, j;
     int a[3][2] = {  { 1, 4 },
                      { 5, 2 },
                      { 6, 5 } };

     for (i = 0; i < 3; i++) {
          for (j = 0; j < 2; j++) {
                    printf("%d ", a[i][j]);
          }
          printf("\n");
     }
     return 0;
}
```

Truyền mảng vào tham số của hàm

- ***Cách 1***: Truyền tham số như con trỏ (sẽ học ở chương sau)

```
void myFunction(int *param) {    ...
}
```

- ***Cách 2:*** Truyền như mảng có kích thước

```
void myFunction(int param[10]) {   ...
}
```

- ***Cách 3:*** Truyền như mảng không có kích thước xác định

```
void myFunction(int param[]) {   ...
}
```

- Với mảng nhiều chiều, ta cái đặt tham số hình thức của hàm tương tự như một chiều, nhưng ít nhất phải xác định số lượng phần tử của chiều cuối cùng trong mảng

```
void myFunction(int param[][10]) {
        ...
}
```

Ví dụ

```c
#include <stdio.h>

/* Khai báo hàm */
double getAverage(int arr[], int size);

int main () {
    /* Một mảng 5 phần tử */
    int balance[5] = {1000, 2, 3, 17, 50};
    double avg;

    /* Truyền mảng như một tham số */
    avg = getAverage( balance, 5 ) ;

    /* Xuất kết quả trả về */
    printf( "Average value is: %f ", avg );
    return 0;
}
```

Một số lỗi thường gặp

```c
#include <stdio.h>

int main () {
        // Khai báo mảng 10 phần tử
        int array[10];
        // Gán giá trị các phần tử
        for (int i=0;i<=10;i++)
                array[i] = i;
        return 0;
}
```

Lỗi do truy xuất vượt quá số lượng phần tử của mảng

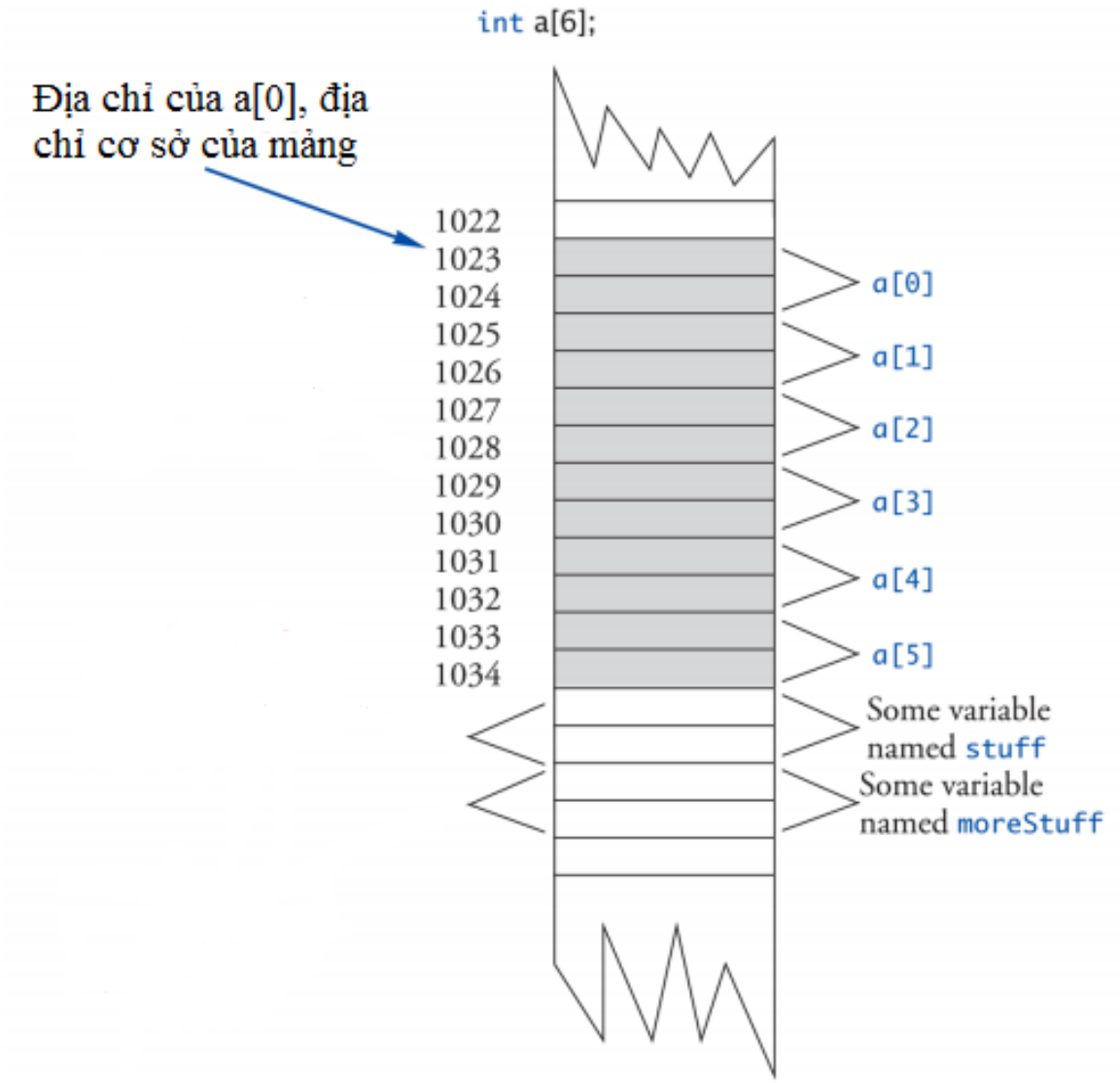
Tổng kết

- Ta dùng mảng như một dạng danh sách đơn giản hoặc để nhóm các dữ liệu có cùng một cách có thứ tự
- Mảng giúp tổ chức và quản lý tập hợp dữ liệu hiệu quả và trực quan hơn.

Bổ sung - Kiểu dữ liệu cơ bản

STT	Kiểu dữ liệu	Dung lượng	Miền giá trị
1	char	1 byte	-128 đến 127
2	int	2 bytes	-32768 đến 32767
3	float	4 bytes	$3,4*10^{-38}$ đến $3,4*10^{38}$
4	double	8 bytes	$1,7*10^{-308}$ đến $1,7*10^{308}$
5	Long double	10 bytes	$3,4*10^{-4932}$ đến $3,4*10^{4932}$

Bổ sung - Cấp phát bộ nhớ cho mảng



Kiểm tra kiến thức

1. Điều gì sẽ xảy ra khi ta gán một phần tử vượt quá chỉ số của mảng?
   a. Phần tử sẽ được gán ở chỉ số 0
   b. Trình biên dịch sẽ báo lỗi
   c. Chương trình có thể bị treo nếu một số dữ liệu bị ghi đè
   d. Kích thước mảng sẽ tự tăng theo

Kiểm tra kiến thức

2. Nếu bạn truyền mảng như một tham số của một hàm, bạn đang truyền
   a. Giá trị của các phần tử trong mảng
   b. Phần tử đầu tiên trong mảng
   c. Địa chỉ cơ sở của mảng
   d. Địa chỉ của phần tử cuối cùng trong mảng

Kiểm tra kiến thức

3. Cho chương trình sau:

```
#include<stdio.h>

int main() {
        int a[5] = {5, 1, 15, 20, 25};

        int i, j, m;
        i = ++a[1];
        j = a[1]++;
        m = a[i++];
        printf("%d, %d, %d", i, j, m);

        return 0;
}
```

a. 2, 1, 15

b. 1, 2, 5

c. 3, 2, 15

d. 2, 3, 20

Kiểm tra kiến thức

4. Cho chương trình sau, cho biết nội dung hiển thị:

```
#include<stdio.h>

int main() {
        int arr[5], i=0;
        while(i<5) arr[i]=++i;

        for(i=0; i<5; i++)
                printf("%d, ", arr[i]);

        return 0;
}
```

a. 1, 2, 3, 4, 5,

b. Giá trị rác, 1, 2, 3, 4,

c. 0, 1, 2, 3, 4,

d. 2, 3, 4, 5, 6,

Kiểm tra kiến thức

5. Cho chương trình sau, cho biết nội dung hiển thị:

```
#include<stdio.h>

int main() {
        int arr[1]={10};
        printf("%d\n", 0[arr]);
        return 0;
}
```

a. 1
b. 10
c. 0
d. 6

Kiểm tra kiến thức

6. Cho chương trình sau, cho biết nội dung hiển thị:

```
#include<stdio.h>

int main() {
        float arr[] = {12.4, 2.3, 4.5, 6.7};
        printf("%d\n", sizeof(arr)/sizeof(arr[0]));
        return 0;
}
```

a. 5
b. 4
c. 6
d. 7

Kiểm tra kiến thức

7. Nếu mảng bắt đầu bằng địa chỉ 1200 trong bộ nhớ, phần nội dung xuất ra của chương trình này là gì

```
#include<stdio.h>

int main() {
        int arr[]={2, 3, 4, 1, 6};
        printf("%d, %d, %d\n", arr, &arr[0], &arr);
        return 0;
}
```

a. 1200, 1202, 1204

b. 1200, 1200, 1200

c. 1200, 1204, 1208

d. 1200, 1202, 1200

Kiểm tra kiến thức

8. Cho chương trình sau, cách cài đặt hàm nào bên dưới là đúng?

```
#include<stdio.h>

int main() {
        int a[3][4];
        fun(a);
        return 0;
}
```

a.
```
void fun(int p[][4]){ ... }
```

b.
```
void fun(int *p[4]){ ... }
```

c.
```
void fun(int *p[][4]){ ... }
```

d.
```
void fun(int *p[3][4]){ ... }
```

Bài tập

1. Viết chương trình cho phép nhập một mảng n các số nguyên dương với n nhập từ bàn phím. Xuất mảng đã nhập ra màn hình
2. Viết chương trình cho phép nhập một mảng n chỉ gồm các số nguyên tố, n nhập từ bàn phím. Nếu nhập không đúng, yêu cầu nhập lại. In mảng.
3. Viết chương trình phân tích một số nguyên n sang hệ nhị phân, lưu trữ các giá trị nhị phân trong một mảng đơn. Xuất kết quả lên màn hình.

Bài tập

4. Viết chương trình cho phép nhập một mảng 2 chiều gồm m dòng, n cột. Giá trị m, n nhập từ bàn phím. Xuất ma trận ra màn hình.
5. Viết chương trình cho phép nhập 2 ma trận 2 chiều m dòng, n cột lần lượt là A và B. Tính ma trận tổng C và xuất ma trận C lên màn hình.

Mời các bạn đặt câu hỏi

CT101 – Lập trình căn bản

Khoa CNTT&TT – ĐHCT