CHƯƠNG 3. KIỂU MẢNG

1. Định nghĩa mảng

Mảng là một tập hữu hạn các phần tử có cùng kiểu dữ liệu và được lưu trữ liên tiếp trong bộ nhớ (RAM). Mảng có thể được xem như là một tập các biến cùng kiểu dữ liệu với nhau. Mỗi phần tử của mảng được xác định và truy xuất bởi chỉ số tương ứng trong mảng. Số phần tử của mảng được gọi là kích thước mảng.

Mảng được sử dụng khi chương trình cần lưu trữ nhiều giá trị có cùng kiểu. Mảng giúp cho việc thao tác trên một tập dữ liệu được tiện lợi hơn. Bên cạnh đó mảng còn được sử dụng để lưu trữ dữ liệu tra cứu nhằm tăng hiệu quả xử lý của thuật toán.

Mảng được chia thành 2 loại: mảng một chiều và mảng nhiều chiều.

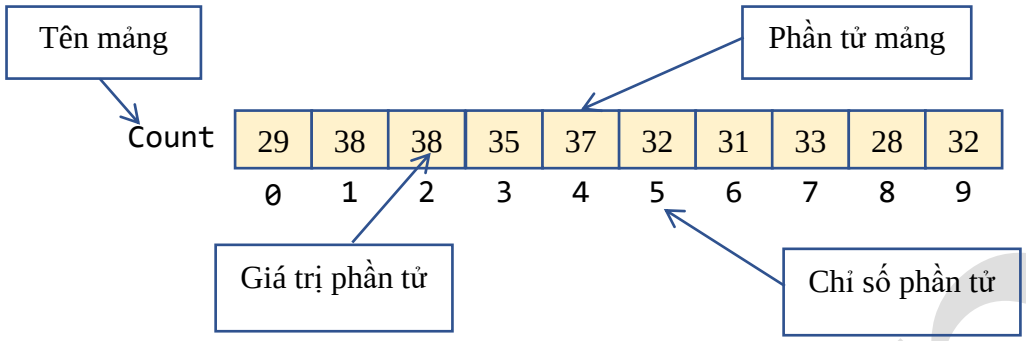
2. Mảng một chiều

Mảng một chiều được sử dụng khi chương trình cần lưu trữ một tập giá trị cùng kiểu có dạng dãy. Các phần tử của mảng được đánh chỉ số bắt đầu từ 0, nghĩa là mảng có kích thước n thì các phần tử được đánh chỉ số từ 0 đến $n-1$.

Xét bảng dữ liệu về danh sách các lớp như sau:

STT	Tên Lớp	Phòng học	Số HS
1	10A1	001	29
2	10A2	002	38
3	10A3	003	38
4	10A4	004	35
5	10A5	005	37
6	10B1	006	32
7	10B2	101	31
8	10B3	102	33
9	10B4	103	28
10	10B5	104	32

Cột dữ liệu về số lượng học sinh của từng lớp trong bảng trên tạo thành mảng một chiều được minh họa như hình bên dưới, danh sách dữ liệu về số lượng học sinh được đặt tên `Count`.



Mỗi ô trong dãy biểu diễn một phần tử của mảng. Phần tử thứ 0 có giá trị 29 cho biết số học sinh của lớp thứ 0 là 29, tương tự phần tử thứ 1 có giá trị 38, …

2.1. Khai báo mảng

Cú pháp: `<kiểu dữ liệu> <tên_mảng>[<kích thước mảng>];`

Lưu ý: Mảng *nên được khai báo toàn cục* khi có kích thước lớn (cỡ 10^3 phần tử trở lên).

Ví dụ 1: Khai báo 2 mảng lưu giá trị 2 dãy số nguyên $a_0, a_1, \dots, a_{n-1}$ và $b_0, b_1, \dots, b_{m-1}$ với $(n, m \leq 10^3)$.

```
int a[1000], b[1000];
```

Ví dụ 2: Khai báo mảng lưu trữ điểm trung bình học kì của các học sinh trong một lớp.

```
double GPA[100];
```

Ví dụ 3: Khai báo mảng số nguyên a có kích thước 1000 và khởi tạo tất cả phần tử của mảng có giá trị 0.

```
int a[1000] = {0};
```

Lưu ý: khi khai báo và khởi tạo như sau `int a[1000] = {10};` thì chỉ có phần tử `a[0]` có giá trị 10, các phần tử còn lại có giá trị 0.

2.2. Truy xuất phần tử mảng

Phần tử của mảng được truy xuất thông qua tên mảng và chỉ số của phần tử tương ứng.

Mỗi phần tử của mảng tương đương với một biến trong chương trình.

Cú pháp: `<tên_mảng>[chỉ_số];`

Ví dụ:

```
a[1] = 10;
cout<<GPA[1];
++b[0];
cout<<b[0];
```

Lưu ý: chỉ số phần tử của mảng phải là một giá trị nguyên thuộc phạm vi từ `0` đến `<kích thước mảng>-1`, nghĩa là `0 ≤ chỉ_số ≤ <kích thước mảng>-1`.

3.3. Một số ví dụ minh họa

Ví dụ 1: Cho số nguyên n và dãy số nguyên $a_0, a_1, \dots, a_{n-1} (n \leq 10^6)$. Tính tổng các phần tử chẵn của dãy.

Input: Dòng đầu chứa số nguyên n. Dòng tiếp theo chứa dãy $a_0, a_1, \dots, a_{n-1}$.

Output: Một số nguyên là tổng tìm được.

```
#include <iostream>

using namespace std;
#define maxN 1000000

int a[maxN], n;

void ReadData()
{
    cin>>n;
    for (int i = 0; i < n; ++i)
        cin>>a[i];
}

int Solve()
{
    int s = 0;
    for (int i = 0; i < n; ++i)
```

Định nghĩa hằng số

```
        if (a[i] % 2 == 0)
            s = s + a[i];
    return s;
}

int main()
{
    ReadData();
    int res = Solve();
    cout<<res;
    return 0;
}
```

Lưu ý: Nên chia các công việc xử lý thành các hàm, mỗi hàm thực hiện một tác vụ độc lập của thuật toán.

Ví dụ 2: Cho 2 số nguyên n, m và 2 dãy số nguyên $a_0, a_1, \dots, a_{n-1} (n \leq 10^6)$ và $b_0, b_1, \dots, b_{m-1} (m \leq 10^6)$. Tính độ lệch giữa 2 phần tử lớn nhất của 2 dãy.

Input:

Dòng đầu chứa hai số nguyên n, m.

Dòng thứ hai chứa dãy $a_1, a_2, \dots, a_n$.

Dòng thứ ba chứa dãy $b_1, b_2, \dots, b_m$.

Output: Một số nguyên là kết quả tìm được.

```
#include <iostream>
#include <cmath>

using namespace std;
#define maxN 1000000

int a[maxN], b[maxN], n, m;

void ReadData()
{
    cin>>n>>m;
    for (int i = 0; i < n; ++i)
        cin>>a[i];
    for (int i = 0; i < m; ++i)
        cin>>b[i];
}
```

Truyền mảng là tham số cho hàm

```
int MaxVal(int a[], int n)
{
    int res = a[0];
    for (int i = 1; i < n; ++i)
        if (res < a[i])
            res = a[i];
    return res;
}

int main()
{
    ReadData();
    cout<<abs(MaxVal(a, n) - MaxVal(b, m));
    return 0;
}
```

Lưu ý:

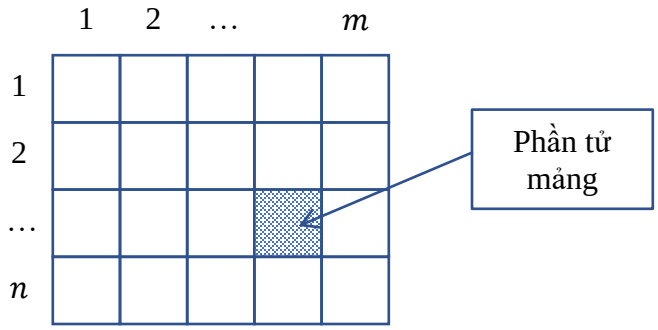
- Nếu sử dụng mảng là tham số cho hàm thì khi khai báo không cần chỉ định kích thước. Khi mảng là tham số của hàm thì mảng có vai trò là tham biến.
- Sử dụng mảng là tham số cho hàm giúp hàm có tính tái sử dụng trong trường hợp thực hiện các thao tác tương tự trên các mảng khác nhau.

4. Mảng 2 chiều

Mảng 2 chiều (bảng) được sử dụng khi chương trình cần lưu trữ một tập dữ liệu có cùng kiểu và có dạng bảng gồm các dòng và cột. Dòng và cột của mảng 2 chiều được đánh chỉ số là các số nguyên liên tiếp bắt đầu từ 0 (zero-base index).

Mảng 2 chiều có thể được xem như là mảng 1 chiều mà mỗi phần tử là một mảng 1 chiều.

Hình ảnh minh họa mảng 2 chiều kích thước $n \times m$ (n dòng, m cột)



Dòng được đánh số thứ tự liên tiếp từ trên xuống và bắt đầu từ 0, cột được đánh thứ tự liên tiếp từ trái qua phải và bắt đầu từ 0.

Mỗi phần tử của bảng tương đương một biến trong chương trình và được xác định thông qua chỉ số dòng, cột của nó trong bảng.

4.1. Khai báo mảng

Cú pháp:

```
<kiểu dữ liệu> <tên_mảng>[<kích thước dòng>][<kích thước cột>];
```

Ví dụ:

Khai báo mảng 2 chiều kích thước $n \times m (n, m \leq 10^3)$ lưu trữ các số nguyên. Dòng được đánh thứ tự từ 0 đến $n-1$, cột được đánh thứ tự từ 0 đến $m-1$.

```
#define maxN 1000
int a[maxN][maxN];
```

Lưu ý: Mảng nhiều chiều có cú pháp khai báo tổng quát như sau

```
<kiểu dữ liệu> <tên_mảng>[<size_1>][<size_2>]…[<size_k>];
```

Trong đó `size_1, size_2, …, size_k` là kích thước của *k* chiều tương ứng.

4.2. Truy xuất phần tử mảng

Phần tử của mảng được truy xuất thông qua tên mảng cùng với cặp chỉ số dòng, cột của phần tử tương ứng.

Cú pháp: `<tên_mảng>[chỉ_số_dòng][chỉ_số_cột];`

Ví dụ:

```
a[1][1] = 10;

cin>>a[i][j];

cout<<a[10][5];
```

4.3. Một số ví dụ minh họa

Ví dụ 1:

Cho bảng số nguyên kích thước $n \times m (n, m \leq 10^3)$. Tìm vị trí của phần tử lớn nhất bảng.

Input: Dòng đầu chứa 2 số nguyên n, m. Dòng thứ i trong n dòng tiếp theo chứa dãy gồm m số nguyên $a_{i1}, a_{i2}, \ldots, a_{im}$ mô tả các phần tử trên dòng thứ i của mảng.

Output: Hai số nguyên x, y tương ứng với chỉ số dòng và chỉ số cột của phần tử lớn nhất bảng. Nếu có nhiều kết quả thì tìm x, y bất kỳ. Chỉ số của mảng được tính từ 1.

```
#include <iostream>

using namespace std;
#define maxN 1000

int a[maxN+1][maxN+1], n, m;

void ReadData()
{
    cin>>n>>m;
    for (int i = 1; i <= n; ++i)
        for (int j = 1; j <= m; ++j)
            cin>>a[i][j];
}

void Solve()
{
    int x = 1, y = 1;
    for (int i = 1; i <= n; ++i)
        for (int j = 1; j <= m; ++j)
            if (a[i][j] > a[x][y])
                x = i, y = j;

    cout<<x<<" "<<y;
}

int main()
{
    ReadData();
    Solve();
    return 0;
}
```

Tăng kích thước thêm 1 vì các chỉ số tính từ 1 trở đi

Câu lệnh kép

Ví dụ 2:

Cho bảng vuông các số nguyên kích thước $n \times n (n \leq 10^3)$. Cho biết các phần tử của bảng có đối xứng qua đường chéo chính hay không. Đường chéo chính của bảng là đường chéo bắt đầu từ ô $(1,1)$ đến ô (n,n).

Input: Dòng đầu chứa số nguyên n. Mỗi dòng trong n dòng tiếp theo chứa dãy gồm n số nguyên mô tả các phần tử trên bảng.

Output: Thông báo YES tương ứng với câu trả lời có, ngược lại thông báo NO.

```
#include <iostream>

using namespace std;
#define maxN 1000

int a[maxN][maxN], n, m;

void ReadData()
{
    cin>>n;
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < n; ++j)
            cin>>a[i][j];
}

bool Solve()
{
    for (int i = 0; i < n; ++i)
        for (int j = 0; j < i; ++j)
            if (a[i][j] != a[j][i])
                return false;

    return true;
}

int main()
{
    ReadData();
    if (Solve())
        cout<<"YES";
    else cout<<"NO";
    return 0;
}
```

Định nghĩa hằng số maxN có giá trị 1000