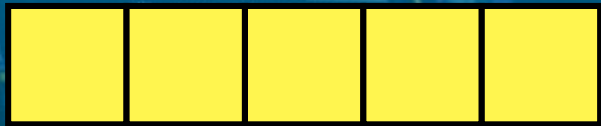




28TECH

Become A Better Developer

MẢNG MỘT CHIỀU (1D ARRAY)



NỘI DUNG



Bài toán cần giải quyết



Định nghĩa và tính chất



Các thao tác trên mảng

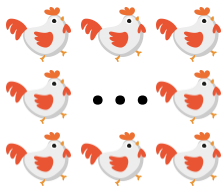


Một số bài toán cơ bản



1. BÀI TOÁN CẦN GIẢI QUYẾT

Bài toán: Giả sử bạn cần tính trọng lượng của 1 đàn gà lên đến hàng nghìn con. Vậy việc lưu trữ trọng lượng của hàng nghìn con gà này sẽ phải xử lý ra sao?



Giải pháp:

Khai báo 1000 biến cho 1000 con gà



Nhiều hạn chế, không phù hợp với **số lượng lớn**

Sử dụng **mảng một chiều**



Hiệu quả, dễ quản lí



2. ĐỊNH NGHĨA VÀ TÍNH CHẤT

Mảng 1 chiều là một **cấu trúc dữ liệu** gồm nhiều phần tử có cùng kiểu dữ liệu, được lưu trữ ở **các ô nhớ liên tiếp** nhau trong bộ nhớ.



Mảng 1 chiều được sử dụng khi bạn cần lưu trữ một số lượng lớn các phần tử có **cùng kiểu dữ liệu**. Ví dụ: 100 số nguyên, 1000 số thực, 1000 học sinh,...

Mảng 1 chiều đơn giản, dễ hiểu và được sử dụng rất nhiều trong mọi ngôn ngữ lập trình.





3. KHAI BÁO MẢNG

SYNTAX

`Data_type` `array_name` [`Number_of_element`]

Ví dụ:

Khai báo	Ý nghĩa
<code>int a[100];</code>	Mảng số nguyên int a có 100 phần tử
<code>float b[1000];</code>	Mảng số thực float b có 1000 phần tử
<code>double diem[10];</code>	Mảng số thực double diem có 10 phần tử
<code>char ten[50];</code>	Mảng kí tự char ten có 50 phần tử



3. KHAI BÁO MẢNG

| Ví dụ 1:

Khai báo mảng a có 3 phần tử là số nguyên, các giá trị của a là **giá trị rác**.

```
int a[3];
```

100

0

2498

3. KHAI BÁO MẢNG

Ví dụ 2:

Khai báo mảng a có 3 phần tử là số nguyên, **gán** lần lượt các phần tử trong mảng là 1,2,3

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

1

2

3

3. KHAI BÁO MẢNG

Ví dụ 3:

Khai báo mảng a có 3 phần tử là số nguyên, **chỉ khởi tạo phần tử đầu tiên**, khi đó mọi phần tử **không được khởi tạo sẽ có giá trị là 0**

```
int a[3] = {100};
```

100

0

0



▶ 3. KHAI BÁO MẢNG

| Ví dụ 3:

Vậy nếu bạn muốn khởi tạo một mảng
toàn số 0. Cú pháp:

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





▶ 4. TRUY CẬP PHẦN TỬ VÀ DUYỆT MẢNG

Các phần tử trong mảng được truy cập **thông qua chỉ số**.

Chỉ số của mảng được đánh từ 0 và kết thúc bởi $n - 1$ (với n là số lượng phần tử của mảng)

Cú pháp truy cập
`array_name [index]`



▶ 4. TRUY CẬP PHẦN TỬ VÀ DUYỆT MẢNG

| Ví dụ:

```
int a [6] = {3, 8, 9, 1, 7, 4};
```

Array

3

8

9

1

7

4

Index

0

1

2

3

4

5

4. TRUY CẬP PHẦN TỬ VÀ DUYỆT MẢNG

| Ví dụ:

$a[0] =$

3

$a[1] =$

8

$a[2] =$

9

$a[3] =$

1

$a[4] =$

7

$a[5] =$

4

4. TRUY CẬP PHẦN TỬ VÀ DUYỆT MẢNG

Ví dụ về nhập và duyệt mảng:

Duyệt thông qua chỉ số:

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n; // số lượng phần tử trong
    mảng
    cin >> n;
    int a[1000]; // Chú ý nếu n lớn phải
    // khai báo mảng a ít nhất n phần tử
    for(int i = 0; i < n; i++){
        cin >> a[i];
    }
    for(int i = 0; i < n; i++){
        cout << a[i] << ' ';
    }
}
```

4. TRUY CẬP PHẦN TỬ VÀ DUYỆT MẢNG

| Ví dụ về nhập và duyệt mảng:

Duyệt bằng for each

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int a[5] = {1, 2, 3, 4, 5};
    for(int ele : a){
        cout << ele << ' ';
    }
}
```

Output : 1 2 3 4 5

4. TRUY CẬP PHẦN TỬ VÀ DUYỆT MẢNG

Ví dụ về nhập và duyệt mảng:

Duyệt bằng for each

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int a[10] = {1, 2, 3, 4, 5};
    for(int ele : a){
        cout << ele << ' ';
    }
}
```

Output :



4. TRUY CẬP PHẦN TỬ VÀ DUYỆT MẢNG

Chú ý : Khi khai báo kích thước của mảng hãy **chú ý tới giới hạn số lượng phần tử tối đa** của đầu bài.

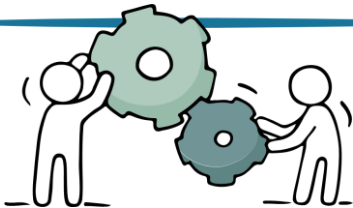
Ví dụ mảng của đầu bài có thể lên đến 1000 thì mảng của bạn phải khai báo tối thiểu là 1000 phần tử.



5. MẢNG LÀM THAM SỐ CỦA HÀM

Khi mảng làm tham số của hàm, những **thay đổi trong hàm** sẽ làm **thay đổi tới mảng mảng** được truyền vào.

Chú ý : Khi xây dựng hàm có tham số là mảng, **cần phải kèm theo 1 tham số nữa là số lượng phần tử** trong mảng



| VD1: Gấp đôi mọi phần tử trong mảng

```
#include <bits/stdc++.h>
using namespace std;

void nhanDoi(int a[], int n){
    for(int i = 0; i < n; i++){
        a[i] *= 2;
    }
}

int main(){
    int a[5] = {1, 2, 3, 4, 5};
    nhanDoi(a, 5);
    for(int i = 0; i < 5; i++){
        cout << a[i] << ' ';
    }
}
```

Output : 2 4 6 8 10

| VD2: Tính tổng các phần tử trong mảng

```
#include <bits/stdc++.h>
using namespace std;

int tong(int a[], int n){
    int sum = 0;
    for(int i = 0; i < n;
i++){
        sum += a[i];
    }
    return sum;
}

int main(){
    int a[5] = {1, 2, 3, 4,
5};
    cout << tong(a, 5) <<
endl;
}

Output : 15
```

6. ƯU VÀ NHƯỢC ĐIỂM CỦA MẢNG

Ưu điểm

Đơn giản, dễ hiểu,
dễ sử dụng

Thuận tiện truy cập
phần tử qua chỉ số,

Khai báo một loạt các phần
tử có cùng kiểu dữ liệu

Duyệt mọi phần tử trong
mảng bằng 1 vòng for

Nhược điểm

Kích thước của mảng là cố
định, bạn không thể mở rộng,
thu hẹp mảng một khi nó đã
được khai báo

Việc chèn và xóa phần tử
trong mảng là khó khăn



7. NO INDEX OUT OF BOUND CHECKING



Mảng trong ngôn ngữ lập trình C++ **không kiểm tra** việc bạn có truy cập vào **một chỉ số hợp lệ hay không**.

Khi mảng của bạn có N phần tử thì chỉ số hợp lệ sẽ là từ 0 tới N - 1. Tuy nhiên bạn **hoàn toàn có thể** truy cập vào các chỉ số không hợp lệ như -1, N, N+1, ...

Các giá trị này **có thể là giá trị rác**, các ô nhớ tương ứng với các phần tử này có thể đang thuộc quản lý của một tiến trình khác.



7. NO INDEX OUT OF BOUND CHECKING

Các ngôn ngữ bậc cao sẽ hạn chế điều này còn trong C++ các bạn lập trình viên phải **tự đảm bảo code của mình không truy cập vào chỉ số không hợp lệ.**

Việc truy cập vào các chỉ số không hợp lệ sẽ gây lỗi **Segmentation Fault** trên **Hackerrank** hoặc các lỗi **Runtime Error** trên các **Online judge** khác.





8. MỘT SỐ BÀI TOÁN CƠ BẢN CẦN LƯU Ý KHI LÀM VIỆC TRÊN MẢNG

1. Tìm phần tử lớn nhất và nhỏ nhất trong mảng.
2. Liệt kê hoặc đếm các phần tử trong mảng thỏa mãn yêu cầu (số nguyên tố, số chính phương, số fibonacci,...)
3. Các bài toán liên quan tới cặp số trong mảng (cặp số nguyên tố cùng nhau, có tổng bằng K,...)
4. Sắp xếp các phần tử trong mảng.
5. Tìm kiếm hoặc đếm số lần xuất hiện của một phần tử nào đó trong mảng.



Tìm phần tử lớn nhất, nhỏ nhất trong mảng

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n; cin >> n;
    int a[n];
    for(int i = 0; i < n; i++) cin >> a[i];
    int max_ele = a[0], min_ele = a[0];
    for(int i = 1; i < n; i++){
        if (a[i] < min_ele){
            min_ele = a[i];
        }
        if (a[i] > max_ele){
            max_ele = a[i];
        }
    }
    cout << min_ele << ' ' << max_ele << endl;
}
```


Tìm phần tử lớn nhất, nhỏ nhất trong mảng

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n; cin >> n;
    int a[n];
    for(int i = 0; i < n; i++) cin >> a[i];
    int max_ele = -1e9, min_ele = 1e9;
    for(int i = 1; i < n; i++){
        max_ele = max(max_ele, a[i]);
        min_ele = min(min_ele, a[i])
    }
    cout << min_ele << ' ' << max_ele << endl;
}
```

Liệt kê hoặc đếm các phần tử trong mảng

Chương trình liệt kê số nguyên tố trong mảng

```
#include <bits/stdc++.h>
using namespace std;

bool nt(int n){
    for(int i = 2; i <= sqrt(n); i++){
        if (n % i == 0) return false;
    }
    return n > 1;
}

int main(){
    int n; cin >> n;
    int a[n];
    for(int i = 0; i < n; i++) cin >> a[i];
    for(int i = 0; i < n; i++){
        if (nt(a[i]))
            cout << a[i] << ' ';
    }
}
```

Các bài toán liên quan tới cặp số trong mảng

Chương trình đếm số cặp trong mảng có tổng bằng K

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n, k;
    cin >> n >> k;
    int a[n];
    for(int i = 0; i < n; i++) cin >> a[i];
    int ans = 0;
    for(int i = 0; i < n; i++){
        for(int j = i + 1; j < n; j++){
            if (a[i] + a[j] == k) {
                ++ans;
            }
        }
    }
    cout << ans << endl;
}
```

Các bài toán liên quan tới cặp số trong mảng

Chương trình đếm số cặp trong mảng có tổng bằng K

Khi bạn cần xét mọi cặp 2 phần tử trong mảng thì bạn cần 2 vòng for lồng nhau.

Tổng quát nếu bạn cần xét mọi cặp k phần tử thì bạn cần k vòng for lồng nhau



Sắp xếp các phần tử trong mảng

Chương trình sắp xếp mảng tăng dần

```
#include <bits/stdc++.h>
#include <algorithm> // For sort
using namespace std;

int main(){
    int n; cin >> n;
    int a[n];
    for(int i = 0; i < n; i++) cin >> a[i];
    sort(a, a + n); // sắp xếp theo thứ tự tăng dần
    for(int i = 0; i < n; i++) cout << a[i] << ' ';
}
```

Input

5

1 3 2 5 4

Output

1 2 3 4 5

Sắp xếp các phần tử trong mảng

Chương trình sắp xếp mảng giảm dần

```
#include <bits/stdc++.h>
#include <algorithm> // For sort
using namespace std;
int main(){
    int n; cin >> n;
    int a[n];
    for(int i = 0; i < n; i++) cin >> a[i];
    sort(a, a + n, greater<int>()); // sắp xếp theo
    // thứ tự giảm dần
    for(int i = 0; i < n; i++) cout << a[i] << ' ';
}
```

Input

5

1 3 2 5 4

Output

5 4 3 2 1

Sắp xếp các phần tử trong mảng

Chương trình sắp xếp mảng giảm dần

Khi sắp xếp giảm dần các bạn thêm tham số thứ 3 là **greater<>**, và chú ý lựa chọn kiểu dữ liệu phù hợp với kiểu dữ liệu trong mảng.

| Ví dụ:

Mảng lưu số **float** thì dùng **greater <float>**.
Tương tự với các kiểu khác như long long, char, ...



Tìm kiếm hoặc đếm số lần xuất hiện của một phần tử nào đó trong mảng

Tìm kiếm sự xuất hiện của phần tử x trong mảng

```
#include <bits/stdc++.h>
#include <algorithm> // For sort
using namespace std;

//ham kiem tra xem x co nam trong mang a hay không?
bool check(int a[], int n, int x){
    for(int i = 0; i < n; i++){
        if(a[i] == x) return true; // found
    }
    return false; // not found
}

int main(){
    int n, x; cin >> n >> x;
    int a[n];
    for(int i = 0; i < n; i++) cin >> a[i];
    if(check(a, n, x)){
        cout << "FOUND\n";
    }
    else cout << "NOT FOUND\n";
}
```


Tìm kiếm hoặc đếm số lần xuất hiện của một phần tử nào đó trong mảng

Đếm số lần xuất hiện của x trong mảng

```
#include <bits/stdc++.h>
#include <algorithm> // For sort
using namespace std;

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

int main(){
    int n, x; cin >> n >> x;
    int a[n];
    for(int i = 0; i < n; i++) cin >> a[i];
    cout << count(a, n, x) << endl;
}
```

