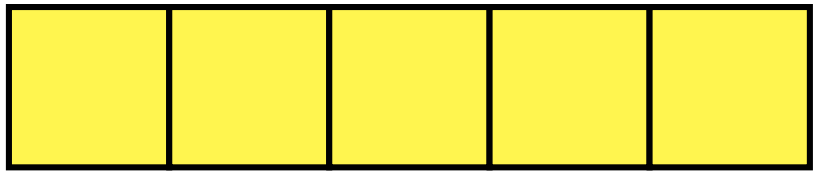
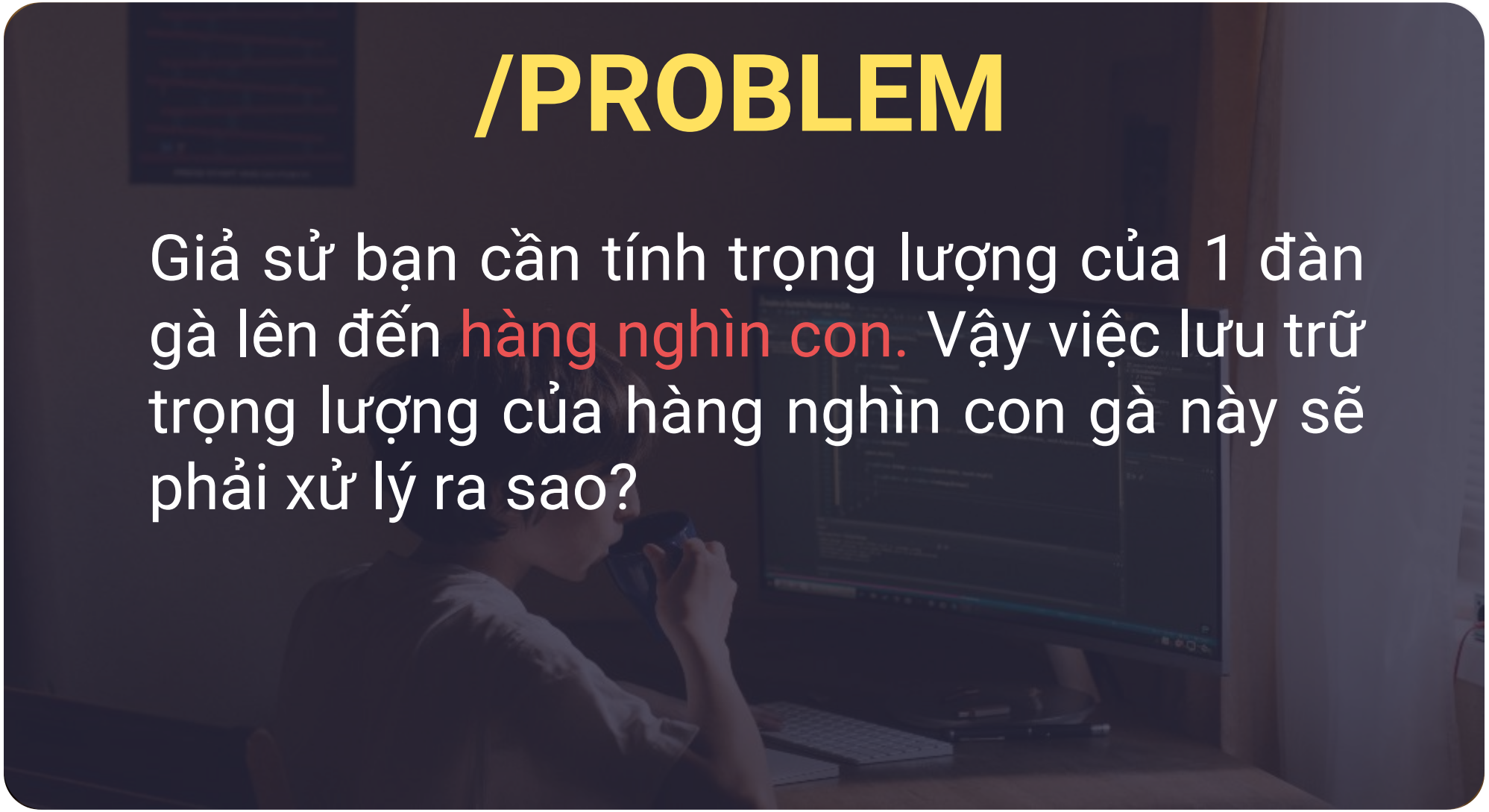
MẢNG MỘT CHIỀU (1D ARRAY)



NỘI DUNG



/PROBLEM

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?

1. ĐỊNH NGHĨA VÀ TÍNH CHẤT

Mảng một 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.

Được sử dụng khi bạn cần lưu trữ một số lượng lớn các phần tử có cùng kiểu dữ liệu.

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

2. KHAI BÁO MẢNG

CÚ PHÁP

```
Data_type array_name [Number_of_element]
```

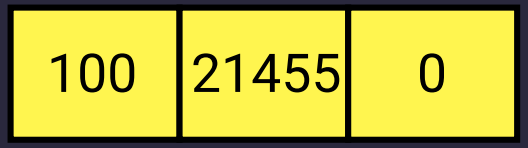
Ví dụ:

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

2. KHAI BÁO MẢNG

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

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



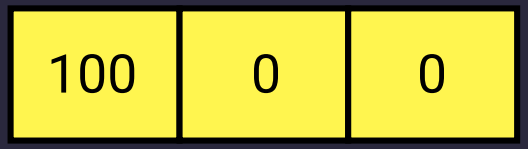
- Khai báo mảng a có 3 phần tử là số nguyên, gán lần lượt các phần tử trong mảng a là 1, 2, 3

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

2. KHAI BÁO MẢNG

- 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};
```



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

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

3. CÁC THAO TÁC TRÊN MẢNG

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]
```

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


```
printf("%d", a[0]); // 3
printf("%d", a[5]); // 4
printf("%d", a[2]); // 9
```

3. CÁC THAO TÁC TRÊN MẢNG

Duyệt mảng thông qua chỉ số

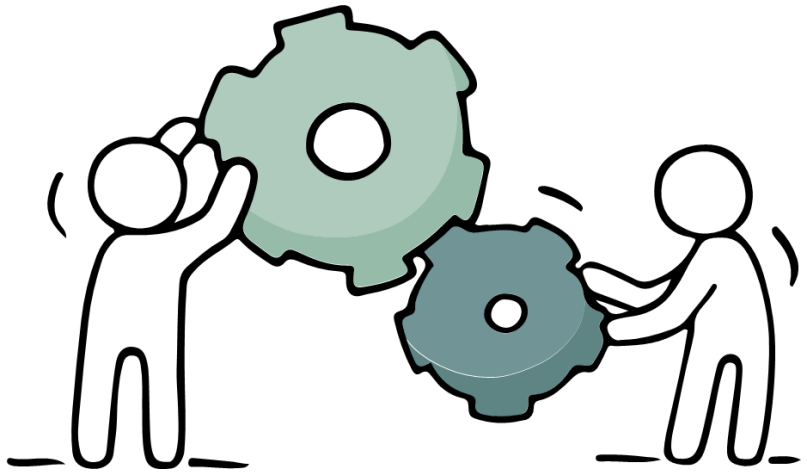
```c
#include <stdio.h>

int main(){
    int n; // Số lượng phần tử mảng
    scanf("%d", &n);
    int a[1000]; //Chú ý nếu n lớn phải
    // khai báo mảng ít nhất n phần tử
    for (int i = 0; i < n; i++){
        scanf("%d", &a[i]);
    }
    for (int i = 0; i < n; i++){
        printf("%d", a[i]);
    }
}
```

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

3. CÁC THAO TÁC TRÊN MẢNG

Mảng làm tham số của hàm



Khi mảng làm tham số của hàm, những thay đổi trong hàm sẽ làm thay đổi tới 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

3. CÁC THAO TÁC TRÊN MẢNG

Ví dụ 1 : Gấp đôi mọi phần tử trong mảng

```
#include <stdio.h>
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++){
                printf("%d", a[i]);
        }
}
```

Output: 2 4 6 8 10

Ví dụ 2 : Tính tổng các phần tử trong mảng

```
#include <stdio.h>
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};
        printf("%d", tong(a, 5));
}
```

Output: 15

4. MỘT SỐ CHÚ Ý TRÊN MẢNG

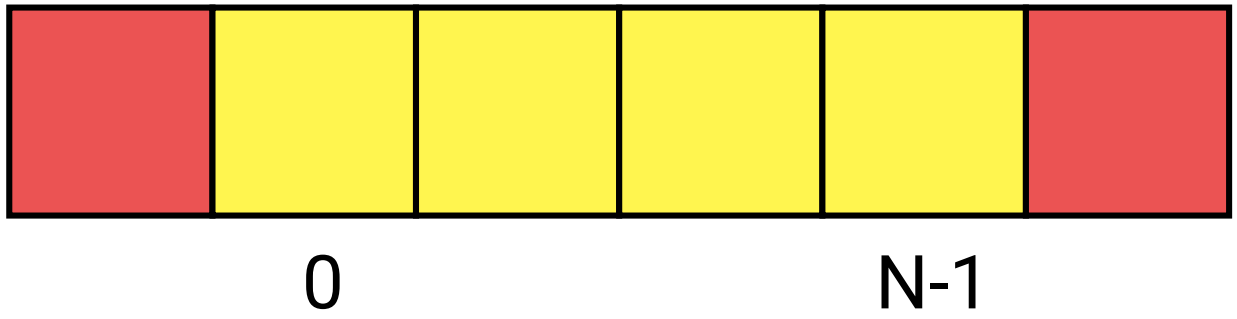
Ưu và nhược điểm của mảng:

- Đơn giản, dễ hiểu và dễ sử dụng
- Truy cập vào phần tử trong mảng nhanh chóng thông qua chỉ số.
- Dễ dàng khai báo một loạt các phần tử cùng kiểu dữ liệu
- Dễ dàng duyệt mọi phần tử trong mảng bằng một vòng lặp duy nhất

- Kích thước của mảng là cố định, bạn không thể mở rộng cũng như thu hẹp mảng một khi nó đã được khai báo
- Việc chèn và xóa phần tử trong mảng là khó khăn

4. MỘT SỐ CHÚ Ý TRÊN MẢNG

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.

4. MỘT SỐ CHÚ Ý TRÊN MẢNG

No index out of bound checking:

Các ngôn ngữ bậc cao sẽ hạn chế điều này còn trong C/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.

5. MỘT SỐ BÀI TOÁN CƠ BẢN

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. 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

Cách 1

```c
#include <stdio.h>

int main(){
    int n; scanf("%d", &n);
    int a[n];
    for(int i = 0; i < n; i++) scanf("%d",&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];
        }
    }
    printf("%d%d", min_ele, max_ele);
}
```

Tìm phần tử lớn nhất nhỏ nhất trong mảng

Cách 2

```c
#include <stdio.h>

int main(){
     int n; scanf("%d", &n);
     int a[n];
     for(int i = 0; i < n; i++) scanf("%d",&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])
    }
    printf("%d%d", min_ele, max_ele);
}
```

Liệt kê hoặc đếm các phần tử trong mảng thỏa yêu cầu

Chương trình liệt kế các số nguyên tố

```
#include <stdio.h>
#include <math.h>
int nt(int n){
     for(int i = 2; i <= sqrt(n); i++){
          if (n % i == 0) return 0;
      }
     return n > 1;
}

int main(){
     int n; scanf("%d", &n);
     int a[n];
     for(int i = 0; i < n; i++) scanf("%d",&a[i]);
     for(int i = 0; i < n; i++){
          if (nt(a[i]))
              printf("%d",a[i]);
     }
}
```

Các bài toán liên quan đến cặp số trong mảng

Chương trình đếm số cặp có tổng bằng K

```
#include <stdio.h>
int main(){
    int n, k; scanf("%d%d", &n, &k);
    int a[n];
    for(int i = 0; i < n; i++) scanf("%d", &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;
       }
    }
    printf("%d", ans);
}
```

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

Tìm kiếm hoặc đếm số lần xuất hiện của 1 phần tử nào đó

Tìm kiếm sự xuất hiện của phần tử x

```
#include <stdio.h>
//ham kiem tra xem x co nam trong mang a hay khong?
int check(int a[], int n, int x){
    for(int i = 0; i < n; i++){
        if(a[i] == x) return 1; // found
    }
    return 0; // not found
}
int main(){
    int n, x; scanf("%d%d", &n, &x);
    int a[n];
    for(int i = 0; i < n; i++) scanf("%d", &a[i]);
    if (check(a, n, x)){
       printf("FOUND\n");
    }
    else printf("NOT FOUND\n");
}
```

Tìm kiếm hoặc đếm số lần xuất hiện của 1 phần tử nào đó

Đếm số lần xuất hiện của phần tử x

```
#include <stdio.h>

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; scanf("%d%d", &n, &x);
    int a[n];
    for(int i = 0; i < n; i++) scanf("%d", &a[i]);
    printf("%d", count(a, n, x));
}
```