Nhập môn Competitive Programming: Mảng (Array)

Ngày 1 tháng 2 năm 2026

Giới thiệu: Mảng (Array)

Chào mừng!

Chào bạn! Rất vui được đồng hành cùng bạn bước chân vào thế giới Competitive Programming (CP).

Chúng ta sẽ cùng khám phá **Mảng (Array)** - một trong những cấu trúc dữ liệu quan trọng nhất.

Tưởng tượng

Hãy coi mảng giống như một **dãy các ngăn kéo tủ** sát nhau. Mỗi ngăn kéo đều có đánh số thứ tự và chứa một món đồ (giá trị) bên trong.

Chunk 1: Mảng là gì & Cách "gọi tên"

Trong lập trình, khi có nhiều dữ liệu cùng loại (ví dụ: điểm của 40 học sinh), thay vì tạo 40 biến, ta dùng một **Mảng**.

- **Chỉ số (Index):** Giống như số nhà. Trong C++, Python, Java, số nhà luôn bắt đầu từ **0**.
- **Giá trị (Value):** Món đồ nằm trong ngăn kéo đó.

Ví dụ

Mảng $A = [10, 25, 30, 45, 50]$

Thử thách tư duy 1

Giả sử mảng A chứa các số: $[10, 25, 30, 45, 50]$.

Câu hỏi

1. Ngăn kéo số 0 (tức là $A[0]$) đang chứa số mấy?
2. Nếu muốn lấy số 45, mình phải mở ngăn kéo số mấy?

Thử thách tư duy 1

Giả sử mảng A chứa các số: $[10, 25, 30, 45, 50]$.

Câu hỏi

1. Ngăn kéo số 0 (tức là $A[0]$) đang chứa số mấy?
2. Nếu muốn lấy số 45, mình phải mở ngăn kéo số mấy?

Đáp án

1. $A[0]$ chính là số **10** (phần tử đầu tiên).
2. Để lấy số **45**: $A[0] = 10, A[1] = 25, A[2] = 30, A[3] = 45$.
   $\rightarrow$ Nó nằm ở ngăn kéo số **3**.

Chunk 2: Duyệt mảng (The Loop)

Duyệt mảng: Đi qua từng ngăn kéo từ đầu đến cuối để kiểm tra.

- Công cụ: Vòng lặp `for`.
- Tương tự: Người đưa thư đi từ nhà số 0, 1, 2... đến hết phố.

Bài toán: Tính tổng

Dùng một biến `tong` (cái rổ) ban đầu bằng 0. Đi qua mỗi ngăn, lấy giá trị cộng vào rổ.

Mã giả:

```
tao bien tong = 0
chay vong lap tu i = 0 den het mang:
    tong = tong + gia tri tai ngan keo thu i
in ra tong
```

Thử thách tư duy 2

Cho mảng $B = [5, 2, 8]$.

Câu hỏi

1. Vòng lặp sẽ chạy qua bao nhiêu "ngôi nhà"(chỉ số)?
2. Tại sao phải đặt biến `tong` $= 0$ ngay từ đầu?

Thử thách tư duy 2

Cho mảng $B = [5, 2, 8]$.

Câu hỏi

1. Vòng lặp sẽ chạy qua bao nhiêu "ngôi nhà"(chỉ số)?
2. Tại sao phải đặt biến `tong` $= 0$ ngay từ đầu?

Giải thích

1. Vòng lặp ghé thăm các nhà số **0, 1, và 2**.
2. Phải đặt `tong` $= 0$ để đảm bảo "cái rổ"trống rỗng. Nếu `tong` $= 10$ từ đầu, kết quả sẽ bị sai (dư 10 đơn vị).

Code minh họa: Tính tổng mảng (C++)

```cpp
#include <iostream>
using namespace std;

int main() {
    // 1. Khai bao mang co 3 phan tu
    int B[] = {5, 2, 8};
    int n = 3; // So luong phan tu

    // 2. Dung "ro" tong de chua ket qua
    int tong = 0;

    // 3. Duyet mang bang vong lap for
    for (int i = 0; i < n; i++) {
        // Cong don gia tri tai ngan keo i vao ro
        tong = tong + B[i];
    }

    cout << "Tong cac phan tu la: " << tong << endl;
    return 0;
}
```

Chunk 3: Thay đổi giá trị & Tìm kiếm

- **Thay đổi:** Mở ngăn kéo, vứt đồ cũ, bỏ đồ mới.
  Ví dụ: `B[1] = 10;` (Mảng $[5, 2, 8] \rightarrow [5, 10, 8]$).
- **Tìm kiếm (Linear Search):** Đi dọc dãy ngăn kéo, hỏi "Đây có phải thứ mình cần không?".

Thử thách tư duy 3

Cho mảng $C = [12, 7, 19, 24]$.

Câu hỏi

1. Cần kiểm tra ít nhất bao nhiêu ngăn kéo để thấy số **7**?
2. Cần kiểm tra bao nhiêu ngăn để biết số **100** KHÔNG có trong mảng?

Thử thách tư duy 3

Cho mảng $C = [12, 7, 19, 24]$.

Câu hỏi

1. Cần kiểm tra ít nhất bao nhiêu ngăn kéo để thấy số **7**?
2. Cần kiểm tra bao nhiêu ngăn để biết số **100** KHÔNG có trong mảng?

Đáp án

1. Cần kiểm tra **2** ngăn (ngăn 0 và 1) để thấy số 7.
2. Cần mở **tất cả 4** ngăn kéo mới dám khẳng định số 100 không tồn tại.

Code minh họa: Tìm kiếm tuyến tính (C++)

```cpp
#include <iostream>
using namespace std;

int main() {
    int C[] = {12, 7, 19, 24};
    int n = 4;
    int x = 100; // So can tim
    bool timThay = false;

    // Duyet qua tung ngan keo de tim x
    for (int i = 0; i < n; i++) {
        if (C[i] == x) {
            timThay = true;
            break; // Thay roi thi dung lai luon
        }
    }

    if (timThay) cout << "Tim thay " << x;
    else cout << "Khong tim thay " << x;

    return 0;
}
```

Chunk 4: Sắp xếp nổi bọt (Bubble Sort)

Tưởng tượng

Các con số là bong bóng dưới nước. Số lớn hơn ("nặng"hơn) sẽ nổi dần về cuối mảng.

Quy tắc: So sánh 2 số cạnh nhau. Nếu số trước **lớn hơn** số sau $\rightarrow$ **Đổi chỗ (Swap)**.

Thử thách tư duy 4

Mảng ban đầu: $[5, 1, 4]$. Sắp xếp tăng dần.

Các bước thực hiện

1. **Lượt 1:** So sánh 5 và 1. Vì $5 > 1$, đổi chỗ.
   Mảng thành: $[1, 5, 4]$.
2. **Lượt 2:** So sánh 5 và 4. Theo bạn, có cần đổi chỗ không? Mảng sẽ thành thế nào?

Thử thách tư duy 4

Mảng ban đầu: $[5, 1, 4]$. Sắp xếp tăng dần.

Các bước thực hiện

1. **Lượt 1:** So sánh 5 và 1. Vì $5 > 1$, đổi chỗ.
   Mảng thành: $[1, 5, 4]$.
2. **Lượt 2:** So sánh 5 và 4. Theo bạn, có cần đổi chỗ không? Mảng sẽ thành thế nào?

Kết quả

Có! Vì $5 > 4$, ta đổi chỗ.
Mảng thành: $[1, 4, 5]$.
Số 5 (lớn nhất) đã "nổi"về đúng vị trí cuối cùng.

Code minh họa: Bubble Sort (C++)

Mã giả:

```
Vong lap i tu 0 den n-2:
    Vong lap j tu 0 den n-i-2:
        Neu A[j] > A[j+1]: Swap(A[j], A[j+1])
```

```cpp
#include <iostream>
#include <algorithm> // Thu vien ham swap
using namespace std;

int main() {
    int a[] = {5, 1, 4};
    int n = 3;

    for (int i = 0; i < n - 1; i++) {
        for (int j = 0; j < n - i - 1; j++) {
            if (a[j] > a[j+1]) {
                swap(a[j], a[j+1]);
            }
        }
    }
    cout << "Mang sau khi sap xep: ";
```

Tổng kết lộ trình

Chúng ta đã hoàn thành các mảnh ghép quan trọng:

- **Truy cập:** Sử dụng chỉ số (Index) bắt đầu từ 0.
- **Duyệt Tính tổng:** Dùng vòng lặp và biến tích lũy.
- **Tìm kiếm:** Linear Search (duyệt tuần tự).
- **Sắp xếp:** Bubble Sort (đổi chỗ các phần tử liền kề).

Lời khuyên

Đây là nền tảng vững chắc để giải các bài tập trên Codeforces. Hãy thực hành viết lại code ngay nhé!