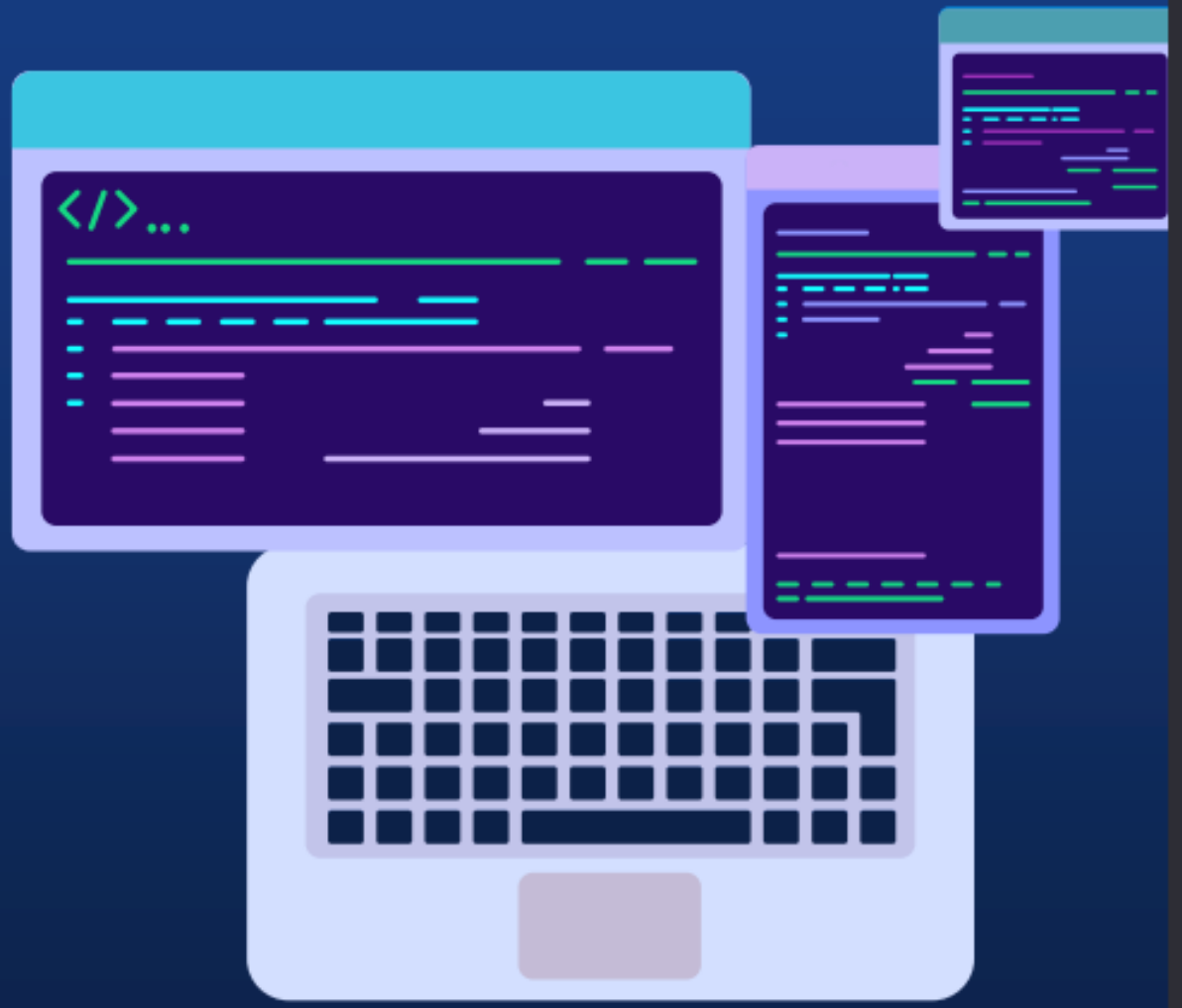
C++ Course



Buổi 5: Mảng 1 chiều, kỹ thuật xử lý
Các bài toán trên mảng 1 chiều

Nguyễn Văn Hiếu

OVERVIEW ARRAY 1D IN C++ LANGUAGE

Mảng 1 chiều là gì, tại sao lại cần sử dụng cấu trúc dữ liệu là mảng 1 chiều:

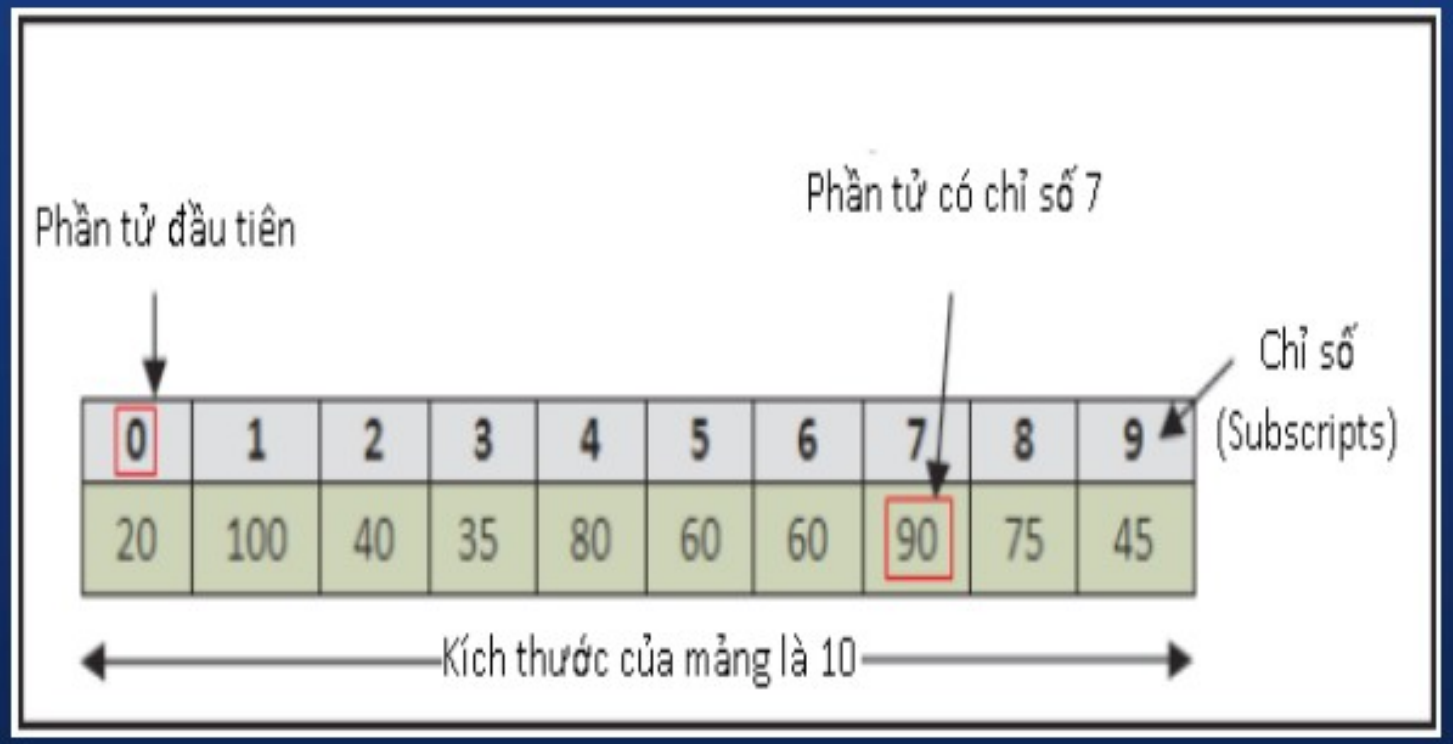
- Lưu trữ một lúc nhiều giá trị có cùng kiểu dữ liệu để xử lý một bài toán lớn.
- Tương tự như tạo ra biến, cần phải có số lượng phần tử để hệ điều hành cấp phát bộ nhớ.
- Kích cỡ của mảng = số phần tử * kích cỡ của một phần tử.
- Các phần tử được xếp liên tiếp nhau trong bộ nhớ máy tính
- **Ví dụ : int array[20]**. Dùng để lưu trữ các giá trị mang kiểu dữ liệu int
- Chú ý về mảng:
  - Trong C++, chúng ta có thể sử dụng các STL:container có sẵn để lưu trữ các dữ liệu tương tự như array với từng nhu cầu và các mục đích khác nhau (cần include header tương ứng vào)
  - Ví dụ : array<int,5> arr = {1,2,3,4,5}
  - Ngoài ra còn chứa vector, set, map, queue, stack, ...
  - Sử dụng các container để dễ dàng thao tác với các container đấy thông qua những thành phần đã được định nghĩa, hỗ trợ sẵn trong C++. (sort() , begin(), end(), ...)
  - Chúng ta sẽ được học các STL:container cùng với việc học OOP

TABLE OF CONTENTS

01

Mảng 1 chiều

Mảng 1 chiều



Hình ảnh về mảng 1 chiều lưu trữ các số nguyên : int a[10]

- Mảng 1 chiều dùng để lưu trữ một dãy các giá trị cùng type
- Mọi mảng bắt đầu từ chỉ số 0, kết thúc ở chỉ số n - 1
- Các phần tử sắp xếp liên tiếp nhau trong bộ nhớ máy tính

Mảng 1 chiều – Ưu và nhược

Ưu điểm

Dùng để lưu trữ một dãy các giá trị cùng loại liên tiếp để xử lý bài toán, khác với một biến đơn thông thường

Có thể được truy cập thông qua chỉ số, thao tác dễ dàng đơn giản

Nhược điểm

Nếu cấp phát không hợp lý có thể gây tốn bộ nhớ, hao phí tài nguyên

Cần phải cung cấp cho hệ điều hành số lượng cố định bộ nhớ, nên có thể thừa, khi thiếu chúng ta sẽ không cấp phát thêm được nữa

Cách sử dụng

Khai báo mảng

Tìm hiểu về kiểu dữ liệu của các phần tử trong mảng để khai báo mảng một cách hợp lý, đầy đủ

Nhập dữ liệu

Thực hiện nhập một dãy dữ liệu đã xác định cho mảng để mảng có thể duyệt dãy dữ liệu và xử lý

Xử lý mảng

Thực hiện duyệt mảng theo cơ chế riêng, thao tác với các phần tử trong bài toán để đưa ra đáp án

Mảng 1 chiều thông thường và STL:Container

Mảng 1 chiều

Do người dùng tự định nghĩa theo nhu cầu, cần cho hệ điều hành biết được lượng bộ nhớ cần cấp phát là bao nhiêu
Không có các method riêng để xử lý sẵn mà chúng ta cần phải thao tác thủ công, hoặc chúng ta tự định nghĩa theo nhu cầu
datatype name[size];

Container

Tất cả đều được định nghĩa sẵn trong các header STL:Container
Thực chất là các class được định nghĩa sẵn, chứa các method mà các attribute sẵn, có iterator riêng để thực hiện thao tác với container đó một cách đơn giản
class<type,size> name;

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a[10];
    for (int i = 0; i < 10; ++i)
    {
        cin >> a[i];
    }
    for (int i = 0; i < 10; ++i)
    {
        cout << a[i] << " ";
    }
    return 0;
}
```

```cpp
#include <iostream>
#include <array>
using namespace std;
int main()
{
    array<int, 10> a;
    for (int i = 0; i < 10; ++i)
    {
        cin >> a[i];
    }
    array<int, 10>::iterator i;
    for (i = a.begin(); i < a.end(); ++i)
    {
        cout << *i << " ";
    }
    return 0;
}
```

02

Sliding window

Nguyễn Văn Hiếu

Sliding window

Kỹ thuật cửa sổ trượt

- Các vấn đề liên quan đến chuỗi tuyến tính, chẳng hạn như mảng, có thể được giải quyết bằng cách sử dụng phương pháp Sliding Window - cửa sổ trượt.
- Một dãy liền kề là một phần của mảng hay được gọi là cửa sổ (Window). Giống như tên của kỹ thuật này, cửa sổ sẽ được trượt trên mảng khi hoàn thành một số thao tác được thực hiện trên các phần tử bên trong. Có nhiều biến thể khác nhau của thuật toán này nhưng chúng ta sẽ chỉ đề cập đến một biến thể chung nhất

Nguyễn Văn Hiếu

Tại sao lại cần học Sliding window

Kỹ thuật xử lý bài toán sử dụng senquence container

- Thuật toán này giúp chúng ta giảm độ phức tạp về thời gian giống như thuật toán Two Pointer.
- Các biến thể của thuật toán này được một phần lớn cộng đồng lập trình sử dụng.
- Chúng ta chỉ đề cập tới vấn đề chung nhất của kỹ thuật này
- **Tiến hành vào ví dụ : Cho một mảng nguyên n phần tử, tính tổng dãy liên tiếp k phần tử, in ra dãy có tổng lớn nhất**

Cách giải quyết

Brute force

- Bây giờ, hãy nói về các phương pháp có thể được sử dụng để giải quyết vấn đề này:
- Brute Force có Độ phức tạp thời gian: O(n2)
- Cách này bản chất rất đơn giản nhưng tốn rất nhiều thời gian.

Sliding window

- Sliding Window có Độ phức tạp thời gian: O(n)
- Đối với cách tiếp cận này, các bạn có thể tự mình thực hiện vì nó rất đơn giản.
- Cùng nhau tìm hiểu cách làm

Sliding window

```
#include <iostream>
using namespace std;
int main()
{
    int a[10];
    int n = 10, k = 3;
    for (int i = 0; i < n; ++i)
    {
        cin >> a[i];
    }
    int sum = 0;
    int max = 0;
    int index = 0;
    for (int i = 0; i < k; ++i)
    {
        sum += a[i];
    }
    for (int i = 1; i < n - k + 1; ++i)
    {
        sum = sum - a[i - 1] + a[i + 2];
        if (max < sum)
        {
            max = sum;
            index = i;
        }
    }
    for (int i = index; i < index + k; ++i)
        cout << a[i] << " ";
    return 0;
}
```

Nguyễn Văn Hiếu

03

Kỹ thuật 2 con trỏ

Nguyễn Văn Hiếu

Kỹ thuật 2 con trỏ

Khái niệm 

 Cách sử dụng

Tác dụng 

 Ví dụ

Kỹ thuật 2 con trỏ là gì ?

- Sử dụng hai con trỏ để di chuyển trên một danh sách/mảng/chuỗi để thực hiện các thao tác tìm kiếm trong một vòng lặp trên cấu trúc dữ liệu.
- **Opposite-Directional:** mỗi con trỏ được đặt ở hai đầu của mảng và chúng di chuyển về phía nhau cho đến khi chúng gặp nhau hoặc thỏa mãn một số điều kiện.
- **Equi-Directional:** mỗi con trỏ đều bắt đầu ở đầu mảng, một con là con trỏ di chuyển chậm trong khi con còn lại là con trỏ nhanh hơn, di chuyển theo cùng một hướng cho đến khi thỏa một điều kiện nhất định nào đó.
- **Tiến hành vào ví dụ : Cho một mảng nguyên n phần tử tăng dần, tìm dãy số có tổng = k;**

Ví dụ:

Cho một dãy số tang dần gồm n phần tử (n = 10)
tìm dãy số có tổng bằng k (k=18)
Ví dụ : 1,2,3,4,5,6,7,8,9,10 => {3,4,5,6}

Sử dụng two pointer để xử lý bài toán

Cách giải quyết

Two Pointer: Chúng ta sẽ sử dụng hai con trỏ trong phương pháp này để khám phá các phần tử trong mảng đã sắp xếp có giá trị mong muốn hay không. Chúng ta bắt đầu bằng cách khởi tạo hai con trỏ, một con trỏ ở chỉ mục ngoài cùng bên trái: start = 0 và con trỏ còn lại ở cuối mảng: end = length (array) -1. Bây giờ chúng ta sẽ lặp lại mảng cho đến khi chỉ mục start < end, sau đó cộng các phần tử được con trỏ tham chiếu để xem chúng có bằng với số được yêu cầu hay không. Chúng ta có thể trả về start và end nếu thỏa mãn yêu cầu. Nếu tổng lớn hơn số yêu cầu, chúng ta sẽ giảm end đi một đơn vị (end -1). Nếu không, chúng ta sẽ tăng start lên một đơn vị (start + 1). Bởi vì mảng đã được sắp xếp nên chúng ta có thể tận dụng lợi thế là nếu chúng ta giảm biến end, số được trỏ bởi nó sẽ nhỏ hơn số phía trước nó, từ đó giảm tổng của hai số. Tương tự khi tăng biến start tổng của hai số sẽ tăng.

Two Pointer

```cpp
#include <iostream>
using namespace std;
int main()
{
    int a[10];
    int n = 5, k = 9;
    for (int i = 0; i < n; ++i)
    {
        cin >> a[i]; // 2 3 6 8 10
    }
    int start = 0, end = n - 1;
    int s;
    while (start < end)
    {
        s = a[start] + a[end];
        if (s < k)
            ++start;
        if (s > k)
            --end;
        if (s == k)
            break;
    }
    for (int i = start; i <= end; ++i)
    {
        cout << a[i] << " ";
    }

    return 0;
}
```

Nguyễn Văn Hiếu

04

Sàng eratosthenes

Sàng nguyên tố

- Một dãy các số nguyên tố được đánh dấu bởi chỉ số là các số nguyên tố.
- Sử dụng CTDL là mảng hoặc container để thực hiện cài đặt
- Do Eratosthenes phát minh
- Tiến hành thực hiện bài toán: Viết chương trình in ra sàng số nguyên tố từ 1->100;

Sàng nguyên tố

```
#include <iostream>
using namespace std;
int main()
{
    int a[101];
    for (int i = 0; i < 101; ++i)
    {
        a[i] = 1;
    }
    a[0] = a[1] = 0;
    int k;
    for (int i = 2; i < 101; ++i)
    {
        k = 2;
        if (a[i] == 1)
        {
            while (i * k <= 100)
            {
                a[i * k] = 0;
                ++k;
            }
        }
    }
    for (int i = 0; i < 101; ++i)
    {
        if (a[i] == 1)
            cout << i << " ";
    }
    return 0;
}
```

Nguyễn Văn Hiếu

05

Luyện tập