CAU TRUC DU LIEU VA GIAI THUAT

**Bài tập.**

Nhập vào một số nguyên dương n, và n số nguyên lần lượt là các phần tử trong dãy a. Hãy đưa ra một số nguyên là tổng tất cả các phần tử trong dãy đó.

**Ví dụ:**

* Test mẫu 1:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 5 1 3 5 -3 0​ | 6 |

* Với n = 5 và a = [1, 3, 5, -3, 0] thì kết quả mong muốn là 6.
* Test mẫu 2:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4 1 2 3 4 | 10 |

* Với n = 4 và a = [1, 2, 3, 4] thì kết quả mong muốn là 10.

**Hướng dẫn đọc dữ liệu.**

Một số bạn sẽ thắc mắc là tại sao trong đề là "Nhập số nguyên n và một dãy", những tại sao trong phần input của Test case lại chỉ có một dãy số mà không có biến n, như vậy thì đọc dữ liệu kiểu gì.

Vấn đề này có thể giải thích đơn giản thế này: Hoàn toàn tồn tại biến n (kích thước của dãy) ở trong input, bạn hãy xem như nó đang được ẩn đi thôi, việc nhập dữ liệu vẫn diễn ra bình thương.

Ví dụ đoạn code nhập dữ liệu:

cin >> n;

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

cin >> a[i];

}

Mọi thắc mắc khác các bạn vui lòng bình luận vào khóa học. Chúc các bạn hoàn thành khóa học một các suôn sẻ.

**Lý thuyết.**

**Cấu trúc dữ liệu mảng là gì ?**

**Mảng (Array)** là một trong các cấu trúc dữ liệu cũ và quan trọng nhất. Mảng có thể lưu giữ một số phần tử cố định và các phần tử này nền có cùng kiểu. Hầu hết các cấu trúc dữ liệu đều sử dụng mảng để triển khai giải thuật. Dưới đây là các khái niệm quan trọng liên quan tới Mảng.

* **Phần tử**: Mỗi mục được lưu giữ trong một mảng được gọi là một phần tử.
* **Chỉ mục (Index)**: Mỗi vị trí của một phần tử trong một mảng có một chỉ mục số được sử dụng để nhận diện phần tử.

Mảng gồm các bản ghi có kiểu giống nhau, có kích thước cố định, mỗi phần tử được xác định bởi chỉ số.

Mảng là cấu trúc dữ liệu được cấp phát liên tục cơ bản.

Ưu điểm của mảng :

* Truy câp phần tử vơi thời gian hằng số **O(1).**
* Sử dụng bộ nhớ hiệu quả.
* Tính cục bộ về bộ nhớ.

Nhược điểm của mảng:

* Không thể thay đổi kích thước của mảng khi chương trình dang thực hiện.

**Mảng động:**

**Mảng động (dynamic aray)** : cấp phát bộ nhớ cho mảng một cách động trong quá trình chạy chương trình trong C là malloc và calloc, trong C++ là new.

Sử dụng mảng động ta bắt đầu với mảng có 1 phần tử, khi số lượng phần tử vượt qua khả năng của ảng thì ta gấp đôi kích thước mảng cũ và copy phần tử mảng cũ vào nửa đầu của mảng mới.

**Ưu điểm:**

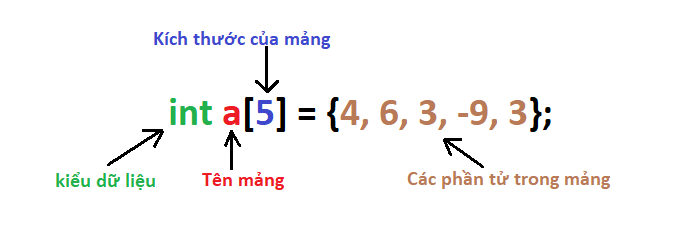
* Tránh lãng phí bộ nhớ khi phải khai báo mảng có kích thước lớn ngay từ đầu.

**Nhược điểm:**

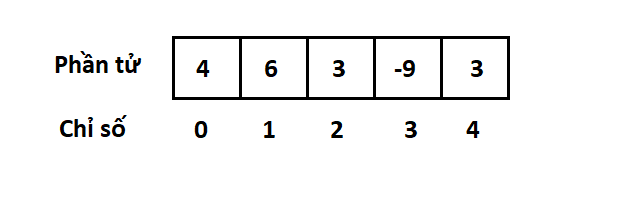
* Phải thực hiện them thao tác copy phần tử mỗi khi thay đổi kích thước.
* Một số thời gian thực hiện thao tác không còn là hằng số nữa.

**Biểu diễn Cấu trúc dữ liệu mảng.**

Mảng có thể được khai báo theo nhiều cách đa dạng trong các ngôn ngữ lập trình. Để minh họa, chúng ta sử dụng phép khai báo mảng trong ngôn ngữ C:



Hình minh họa phần tử và chỉ số:



Dưới đây là một số điểm cần ghi nhớ về cấu trúc dữ liệu mảng:

* Chỉ mục bắt đầu với 0.
* Độ dài mảng là 5, nghĩa là mảng có thể lưu giữ 5 phần tử.
* Mỗi phần tử đều có thể được truy cập thông qua chỉ số của phần tử đó. Ví dụ, chúng ta có thể lấy giá trị của phần tử tại chỉ số 3 là -9.

Phép toán cơ bản được hỗ trợ bởi mảng:

Dưới đây là các hoạt động cơ bản được hỗ trợ bởi một mảng:

* **Duyệt**: In tất cả các phần tử mảng theo cách in từng phần tử một.
* **Chèn**: Thêm một phần tử vào mảng tại chỉ mục đã cho.
* **Xóa**: Xóa một phần tử từ mảng tại chỉ mục đã cho.
* **Tìm kiếm**: Tìm kiếm một phần tử bởi sử dụng chỉ mục hay bởi giá trị.
* **Cập nhật**: Cập nhật giá trị một phần tử tại chỉ mục nào đó.

**Hướng dẫn bài tập.**

Code mẫu:

Ngôn ngữ C:

#include<stdio.h>

int main()

{

int a[101], n;

int Sum = 0;

scanf("%d", &n);

for(int i = 0; i < n; i++)

{

scanf("%d", &a[i]);

Sum += a[i];

}

printf("%d", Sum);

}

Ngôn ngữ C++:

#include<iostream>

using namespace std;

int main(){

int a[100];

int n;

cin >> n;

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

cin >> a[i];

}

int sum = 0;

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

sum += a[i];

}

cout << sum;

return 0;

}

Ngôn ngữ Java:

import java.util.Scanner;

public class B1 {

public static void main(String[] args) {

int n;

int a[] = new int[101];

Scanner scanner = new Scanner(System.in);

n = scanner.nextInt(); //Lệnh nhập giá trị của n từ bàn phím.

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

a[i] = scanner.nextInt(); // Nhập các phần tử trong dãy.

}

int Sum = 0;

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

Sum += a[i];

}

System.out.print(Sum);

}

}

Ngôn ngữ Python:

n = (int)(input())

a = []

for i in range(n):

a.append((int)(input()))

Sum = 0

for i in range(n):

Sum += a[i]

print(Sum)

Ngôn ngữ C#:

using System;

namespace B1{

class Program{

public static void Main(){

int n = int.Parse(Console.ReadLine());

int[] a = new int[n];

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

a[i] = int.Parse(Console.ReadLine());

}

int Sum = 0;

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

Sum += a[i];

}

Console.WriteLine(Sum);

}

}

}

**Bài tập.**

Nhập vào một số nguyên dương n, và n số nguyên lần lượt là các phần tử trong dãy a.  
Hãy thay đổi giá trị của mỗi phần tử thành bình phương của chính nó, sau đó in mảng đó ra màn hình (mỗi phần tử cách nhau bởi một khoảng trắng).

**Ví dụ:**

* Test mẫu 1:

|  |  |
| --- | --- |
| **Input** | **Input** |
| 4 1 3 5 -3 | 1 9 25 9 |

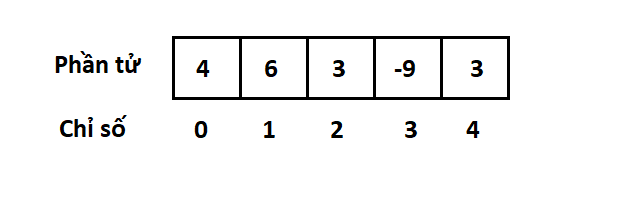
* Với n = 4 và a = [1, 3, 5, -3] thì kết quả mong muốn là: "1 9 25 9 ".
* Test mẫu 2:

|  |  |
| --- | --- |
| **Input** | **Input** |
| 4 1 2 3 4 | 1 4 9 16 |

* Với n = 4 và a = [1, 2, 3, 4] thì kết quả mong muốn là: "1 4 9 16 ".

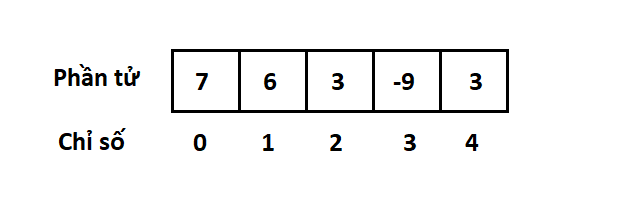
**Hướng dẫn bài tập.**

Để xử lý một phần tử trong mảng ta chỉ cần biết chỉ số của phần tử đó:



Ví dụ a[0] = 4, a[1] = 6, ...

Để cập nhập phần tử ta cũng có thể gán thẳng giá trị cần cập nhật vào cho vị trí đó.  
Ví dụ a[0] = 7 thì dãy được cập nhật là:



Với bài tập trên chúng ta chỉ cần duyệt hết tất cả những phần tử và thay đổi a[i] = a[i]\*a[i] là được.

Code mẫu:

Ngôn ngữ C++:

#include<iostream>

using namespace std;

int main(){

int a[100];

int n;

cin >> n;

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

cin >> a[i];

}

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

a[i] = a[i]\*a[i];

}

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

cout << a[i] << " ";

}

return 0;

}

**Bài tập.**

Nhập vào một số nguyên dương n, và n số nguyên lần lượt là các phần tử trong dãy a. Tiếp theo nhập vào hai số nguyên là k và x (0 ≤ k ≤ n).

Hãy chèn phần tử có giá trị x vào dãy a, ở trước phần tử có chỉ số k và sau phần tử có chỉ số k-1. In mảng kết quả ra màn hình, mỗi số cách nhau bởi một khoảng trắng.

**Ví dụ:**

* Test mẫu 1:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 4 1 2 3 4 1 10 | 1 10 2 3 4 |

* Với n = 4, a = [1, 2, 3, 4], k = 1, x = 10 thì kết quả mong muốn là: "1 10 2 3 4 ".
* Test mẫu 2:

|  |  |
| --- | --- |
| **Input** | **Output** |
| 3 1 2 3 3 123 | 1 2 3 123 |

* Với n = 3, a = [1, 2, 3], k = 3, x = 123 thì kết quả mong muốn là: "1 2 3 123 ".

**Hướng dẫn bài tập.**

Mỗi phần tử trong mảng đều có một chỉ số cố định, để chèn một phần tử x vào trước phần tử có chỉ số k cho trước không phải cứ gán a[k] = x là được, vì nếu gán về thì phần tử a[k] cũ sẽ bị mất.

Ý tưởng cho việc chèn phần tử trong mảng như sau:

Ta thấy rằng sau khi chèn vào dãy một phần tử thì chắc chắn số lượng phần tử sẽ tăng lên 1 đơn vị, các phần tử có chỉ số từ 0 đến k-1 vẫn giữ nguyên, còn các phần tử từ chỉ số k đến n-1 sẽ tăng chỉ số lên 1, để làm được như vậy ta phải gán a[i] = a[i-1] và phải chọn vòng lặp cho hợp lý.

Nhiều bạn đã thử biến đổi như sau:

for (int i = k+1; i <= n; i++){

a[i] = a[i-1];

}

a[k] = x;

n++;

Nhưng kết quả sẽ cho ra là sai, bởi vì các giá trị cũ bị mất đi, ví dụ như sau: nếu gán a[4] = a[3] sau đó là gán a[5] = a[4] thì giá trị a[4] cũ không còn nữa mà nó đang giữ giá trị của a[3].

Để giải quyết vấn đề trên ta sẽ thay đổi vòng for thành như sau:

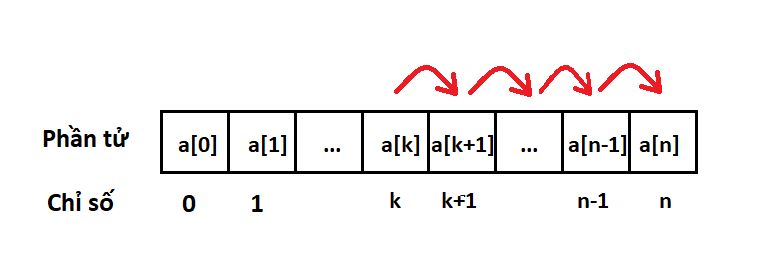
for (int i = n; i >= k+1; i--){

a[i] = a[i-1];

}

a[k] = x;

n++;



Code mẫu:

Ngôn ngữ C++:

#include<iostream>

using namespace std;

int main(){

int a[100];

int n, k, x;

cin >> n;

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

cin >> a[i];

}

cin >> k >> x;

for (int i = n; i >= k+1; i--){

a[i] = a[i-1];

}

a[k] = x;

n++;

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

cout << a[i] << " ";

}

return 0;

}