Question 1

Not answered

Mark 0.00 out of 10.00

[OneDimensionArray]

Trong bài tập ở phần 2 ta đã được học về biến và các thao tác trên biến. Tuy nhiên, có một vấn đề được đặt ra là khi số lượng biến cần phải lưu trở nên quá nhiều, làm thế nào để tổ chức và quản lý các biến này một cách hiệu quả.

Ví dụ, ở một nhà máy có 20 công nhân, mỗi ngày, một công nhân có thể làm ra được một số lượng sản phẩm khác nhau. Quản lý của nhà máy muốn lưu trữ số lượng sản phẩn mà mỗi công nhân làm được trong một ngày để dễ dàng quản lý.

Với bài toán này, ta cần 20 biến để lưu lại số lượng sản phẩm làm được bởi từng công nhân:

```
int so_san_pham1;
int so_san_pham2;
...
int so_san_pham20;
```

Và ta cần 20 dòng lệnh để đọc các biến này từ đầu vào:

```
cin >> so_san_pham1;
cin >> so_san_pham2;
...
cin >> so_san_pham20;
```

Việc khai báo và đọc các biến theo cách này rất phức tạp, chưa kể đến lúc xử lý trên các biến này. Vì vậy, trong ngôn ngữ lập trình C++, có một khái niệm về tổ chức dữ liệu liên tiếp nhau trên thiết bị cung cấp bộ nhớ được gọi là mảng.

Trong C++, có 2 cách khai báo mảng:

Mảng tĩnh

```
<tên kiểu phần tử> <tên mảng>[<số phần tử>];
```

Với cách khai báo này, chúng ta cần ghi rõ cho trình biên dịch biết số lượng phần tử mà bạn cần sử dụng đặt trong cặp dấu ngoặc vuông. Ví dụ,

```
int so_san_pham[20];
```

Mình vừa khai báo một mảng gồm 20 phần tử để lưu số lượng sản phẩm mà các công nhân làm được trong một ngày.

Các phần tử trong mảng được đánh số từ 0:

- so_san_pham[0] là phần tử thứ nhất;
- so_san_pham[1] là phần tử thứ hai;
- ...
- so_san_pham[19] là phần tử thứ hai mươi.

Để đọc thông tin từ đầu vào, ta chỉ cần sử dụng một vòng lặp for đơn giản như sau:

```
for (int i = 0; i < 20; ++i)
    cin >> so_san_pham[i];
```

Như bạn có thể thấy, cách làm này ngắn gọn và đơn giản hơn rất nhiều so với khi ta không sử dụng mảng như ở trên.

Vector

Ngoài cách sử dụng mảng tĩnh như trên, ta còn có thể khai báo mảng động bằng thư viện `<vector>` của C++:

```
vector < <tên kiểu phần tử> > <tên mảng>(<số phần tử>, <giá trị khởi tạo>);
```

Lưu ý, trước khi khai báo một vector ta phải thêm thư viên `<vector>` bằng câu lệnh:

```
#include <vector>
```

Ví dụ:

```
vector <int> so_san_pham(20,0); // Khai báo một vector có 20 phần tử có giá trị bằng 0.
vector <int> so_san_phan(20); // Khai báo một vector có 20 phần tử có giá trị ngẫu nhiên.
```

Cách truy xuất các phần tử trong một vector tương tự cách truy xuất các phần tử trong mảng tĩnh.

Lưu ý: Ta có thể thay đổi kích thước một vector bằng lệnh `resize()`. Ví dụ:

```
so_san_pham.resize(10);
```

Sau câu lệnh trên, vector `so_san_pham` biến thành một vector có 10 phần tử.

Bài tập

Một nhà máy có n công nhân, trong một ngày, mỗi công nhân làm được a_i sản phẩm. Tìm các công nhân làm ra số sản phẩm lớn hơn hoặc bằng trung bình số sản phẩm mà một công nhân làm được trong nhà máy đó.

Ví dụ, một nhà máy có 3 công nhân. Công nhân thứ nhất làm được 3 sản phẩm một ngày, công nhân thứ hai làm được 8 sản phẩm một ngày, công nhân thứ ba làm được 4 sản phẩm một ngày. Trung bình số sản phẩm một công nhân làm được là $\frac{3+8+4}{3} = 5$. Như vậy chỉ có công nhân thứ ba làm ra số sản phẩm lớn hơn hoặc bằng trung bình số sản phẩm mà một công nhân làm được trong nhà máy đó.

Đầu vào

Đầu vào từ bàn phím gồm 2 dòng:

- Dòng đầu tiên chứa số nguyên n $(n \leq 30)$.
- Dòng thứ 2 chứa n số nguyên dương có giá trị không vượt quá 100, cách nhau bởi một dấu cách là số lượng sản phẩm từng công nhân làm được trong một ngày.

Đầu ra

In ra màn hình một dòng chứa tất cả các số nguyên là số sản phẩm của các công nhân làm ra số sản phẩm lớn hơn hoặc bằng mức trung bình, mỗi số cách nhau bởi một dấu cách.

For example:

Input	Result
4 9 2 3 4	9

Answer:

```
1
```

	Input	Expected	Got	
✓	4 9 2 3 4	9	9	✓
✓	12 22 34 35 5 43 12 23 9 32 3 91 27	34 35 43 32 91	34 35 43 32 91	✓
✓	1 12	12	12	✓

Passed all tests! ✓

Back to Course