BÀI TẬP LẬP TRÌNH

SÁCH LƯU HÀNH NỘI BỘ

© Trần Việt Khoa[†]
Second Author[†]

[†] Đại học Khoa học Huế

BÀI TẬP LẬP TRÌNH

Trần Việt Khoa Second Author

Ngày 19 tháng 9 năm 2023

Mục lục

Phần I

Nhập môn lập trình

Chương 1

Tổng quan về lập trình

1.1 Tóm tắt lý thuyết

1.1.1 Khái niệm thuật toán

Một bài toán được giải thì việc đầu tiên phải làm là xây dựng một mô hình dịch bài toán đó thành ngữ cảnh toán học. Các cấu trúc rời rạc được dùng trong các mô hình này là số, ký tự, tập hợp, dãy, hàm, hoán vị, quan hệ, cùng với các cấu trúc khác như đồ thị, cây, mạng.

Lập được một mô hình toán học thích hợp chỉ là một phần của quá trình giải. Để hoàn tất quá trình giải, còn cần phải có một phương pháp dùng mô hình để giải bài toán tổng quát. Nói một cách lý tưởng, cái được đòi hỏi là một thủ tục, đó là dãy các bước dẫn tới đáp số mong muốn. Một dãy các bước như vậy, được gọi là một thuật toán.

Khi thiết kế và cài đặt một phần mềm tin học cho một vấn đề nào đó, ta cần phải đưa ra phương pháp giải quyết mà thực chất đó là thuật toán giải quyết vấn đề này. Rõ ràng rằng, nếu không tìm được một phương pháp giải quyết thì không thể lập trình được. Chính vì thế, thuật toán là khái niệm nền tảng của hầu hết các lĩnh vực của tin học.

Định nghĩa: Thuật toán là một bảng liệt kê các chỉ dẫn (hay quy tắc) cần thực hiện theo từng bước xác định nhằm giải một bài toán đã cho.

Thuật ngữ "Algorithm" (thuật toán) là xuất phát từ tên nhà toán học Ả Rập Al-Khowarizmi. Ban đầu, từ algorism được dùng để chỉ các quy tắc thực hiện các phép tính số học trên các số thập phân. Sau đó, algorism chuyển thành algorithm vào thế kỷ 19. Với sự quan tâm ngày càng tăng đối với các máy tính, khái niệm thuật toán đã được cho một ý nghĩa chung hơn, bao hàm cả các thủ tục xác định để giải các bài toán, chứ không phải chỉ là thủ tục để thực hiện các phép tính số học.

Ví dụ 1.1. Mô tả thuật toán tìm phần tử lớn nhất trong một dãy hữu hạn các số nguyên

Đầu vào: Dãy số nguyên $A = \{a_1, a_2, \ldots, a_n\}$.

Đầu ra: Giá trị lớn nhất.

Thuật toán: Sử dụng ngôn ngữ tự nhiên, ta mô tả như sau:

Bước 1. Đặt giá trị cực đại tạm thời bằng số nguyên đầu tiên trong dãy (Cực đại tạm thời sẽ là số nguyên lớn nhất được kiểm tra ở các bước lặp tiếp theo của thuật toán).

Bước 2. So sánh số nguyên tiếp theo sau với giá trị cực đại tạm thời, nếu nó lớn hơn giá trị cực đại tạm thời thì đặt giá trị cực đại tạm thời bằng số nguyên đó.

Bước 3. Lặp lại bước 2 nếu còn các số nguyên trong dãy chưa so sánh.

Bước 4. Dừng khi không còn số nguyên nào cần so sánh, cực đại tạm thời chính là số lớn nhất của dãy.

Các đặc trưng của thuật toán:

- **Đầu vào (Input)**: Một thuật toán có các giá trị đầu vào từ một tập đã được chỉ rõ. Cụ thể là mô tả rõ kiểu dữ liệu của dữ liệu vào, miền xác định của dữ liệu.
- **Đầu ra (Output)**: Từ mỗi tập các giá trị đầu vào, thuật toán sẽ tạo ra các giá trị đầu ra. Các giá trị đầu ra chính là nghiệm của bài toán.
- **Tính dừng**: Sau một số hữu hạn bước thuật toán phải dừng. Các bài toán đưa ra thuật toán phải chạy được trên máy tính với thời gian chấp nhận được.
- **Tính xác định**: Ở mỗi bước, các bước thao tác phải hết sức rõ ràng, không gây nên sựnhập nhằng. Nói rõ hơn, trong cùng một điều kiện hai bộ xử lý cùng thực hiện một bước của thuật toán phải cho những kết quả như nhau. Sự nhập nhằng thường xảy ra khi ta lẫn lộn giữa kiểu dữ liệu của bài toán, nhầm lẫn giữa các phép toán ví dụ như phép chia, phép so sánh và phép gán chẳng hạn.
- **Tính hiệu quả**: Trước hết thuật toán cần đúng đắn, nghĩa là sau khi đưa dữ liệu vào thuật toán hoạt động và đưa ra kết quả như ý muốn. Tính hiệu quả được đề cập bởi hiệu quả về tài nguyên và hiệu quả về thời gian chạy. Yếu tố thời gian được đánh giá cao hơn.
- **Tính phổ dụng**: Thuật toán có thể giải bất kỳ một bài toán nào trong lớp các bài toán. Cụ thể là thuật toán có thể có các đầu vào là các bộ dữ liệu khác nhau trong một miền xác định.

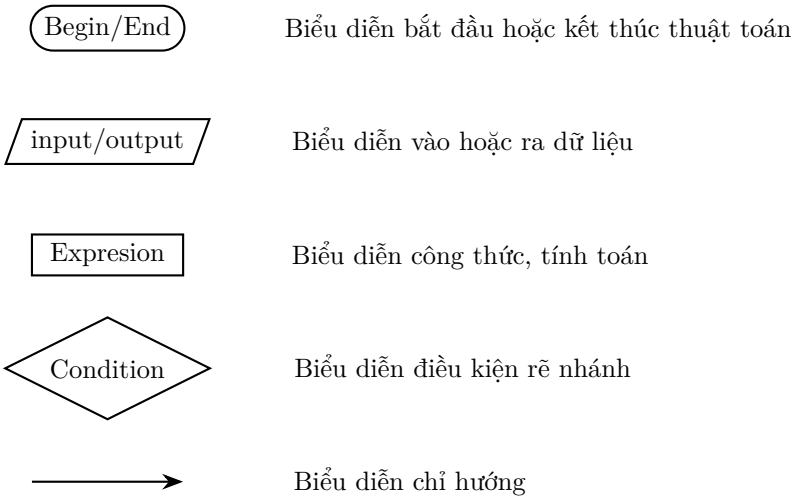
1.1.2 Biểu diễn thuật toán bằng sơ đồ khối

Có nhiều cách trình bày thuật toán:

- Dùng ngôn ngữ tự nhiên, tuy rất dễ trình bày nhưng không có tính phổ dụng.
- Dùng ngôn ngữ giả mã (pseuducode) là cách trình bày thuật toán tốt nhất, được viết theo ngôn ngữ tựa như ngôn ngữ lập trình (có cú pháp và ngữ pháp, tham khảo https://en.wikipedia.org/wiki/Pseudocode) và dễ dàng chuyển nó thành ngôn ngữ lập trình mà người sử dụng muốn.
- Dùng sơ đồ khối (flowchart, tham khảo https://en.wikipedia.org/wiki/Flowchart) là một công cụ có tính thể hiện về mặt hình học rất rõ ràng và hữu ích với người mới học thuật toán.

Sơ đồ khối là một ngôn ngữ đồ họa, chỉ bao gồm một số hình nhưng cũng đủ để trình bày được một thuật toán với quy tắc là mở đầu sơ đồ phải là hình khối **Bắt đầu** và kết thúc sơ đồ phải là hình khối

Kết thúc, tất cả các khối đều phải liên kết với nhau bằng dấu mũi tên, các hình họa gồm:



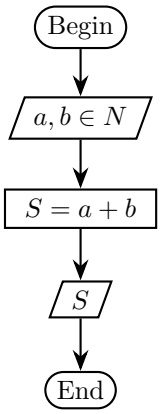
Hình 1.1: Các hình họa của sơ đồ khối

Và chỉ với 5 hình trên đủ để biểu diễn các thuật toán cơ bản.

Ví dụ 1.2. Tính tổng hai số nguyên

Vẽ sơ đồ thuật toán tính tổng hai số nguyên a, b cho trước.

Lời giải:



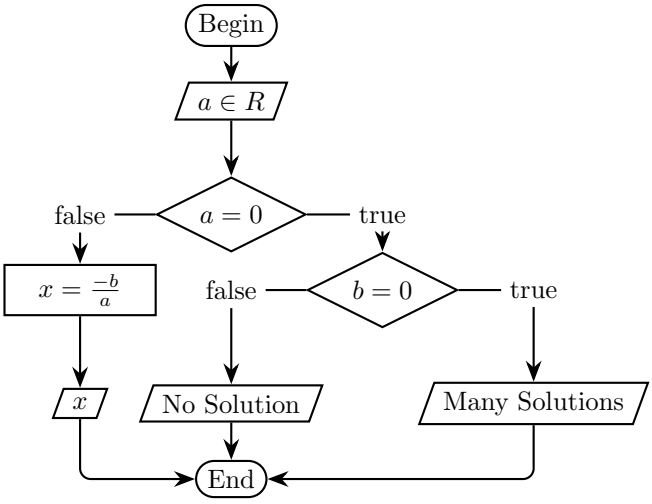
Hình 1.2: Sơ đồ khối tính tổng hai số

Ví dụ 1.3. Giải phương trình bậc nhất

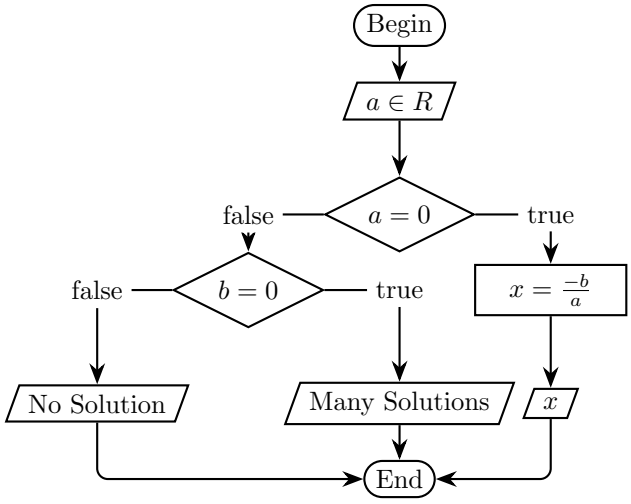
Vẽ sơ đồ thuật toán giải phương trình $ax + b = 0$ với a, b là hai số thực cho trước. In ra nghiệm cần tìm, trường hợp không có nghiệm thì in **No Solution** hoặc trường hợp có nhiều nghiệm thì in **Many Solutions**.

Lời giải:

Bài toán này có thể thiết kế theo hai sơ đồ khối như sau, tùy theo tư duy logic của người học. Cả hai đều đúng về mặt logic nhưng cách vẽ theo bố trí 2 quản lý khối lệnh tốt hơn (có thể tham khảo lệnh



(a) Bố trí 1



(b) Bố trí 2

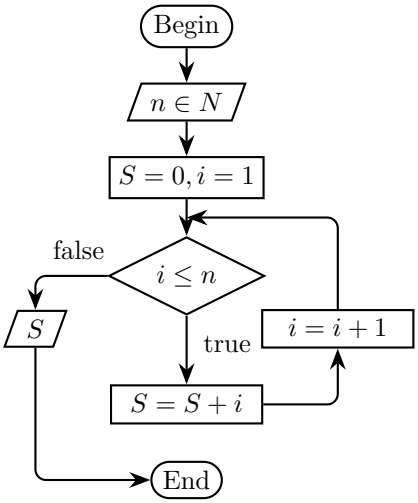
Hình 1.3: Sơ đồ thuật toán giải phương trình $ax + b = 0$.

if..elif..else của ngôn ngữ python).

Ví dụ 1.4. Tính tổng S

Vẽ sơ đồ thuật toán tính giá trị $S = 1 + 2 + \ldots + n$ với n nguyên dương cho trước.

Lời giải:



Hình 1.4: Sơ đồ thuật toán tính $S = 1 + 2 + \ldots + n$

Ưu điểm của sơ đồ:

- Sơ đồ là cách tốt hơn để truyền đạt logic của thuật toán đối với người mới học.
- Rất dễ hiểu khi đọc sơ đồ.
- Dễ dàng chuyển sang ngôn ngữ lập trình.
- Dễ sửa lỗi.

Nhược điểm của sơ đồ khối:

- Rất khó để vẽ sơ đồ cho các chương trình lớn và phức tạp.
- Việc tạo một sơ đồ là tốn kém.
- Nếu thay đổi thuật toán thì phải vẽ lại sơ đồ.

1.1.3 Chương trình máy tính

Chương trình máy tính là tập hợp các chỉ dẫn được thể hiện dưới dạng các lệnh, các mã, lược đồ hoặc bất kỳ dạng nào khác, khi gắn vào một phương tiện mà máy tính đọc được, có khả năng làm cho máy tính thực hiện được một công việc hoặc đạt được một kết quả cụ thể.

Việc xây dựng một chương trình thông dụng nhất bằng cách sử dụng các ngôn ngữ lập trình bậc cao mà ở đó có đầy đủ các lệnh, cấu trúc lệnh và các cơ chế cho phép người lập trình mô tả, lập trình bài toán cần giải quyết trên máy tính.

Lập trình là việc sử dụng cấu trúc dữ liệu và các lệnh của ngôn ngữ lập trình cụ thể (thường là bậc cao) để mô tả dữ liệu và diễn đạt các thao tác của thuật toán cho bài toán cần giải.

Một chương trình thường có các yếu tố sau:

Dữ liệu: Đọc từ bàn phím, từ một tập tin hoặc từ một vài thiết bị khác.

Kết quả: Hiển thị ra màn hình, gửi ra tập tin hoặc các thiết bị khác.

Toán học: Thực hiện các phép toán toán học hoặc logic.

Rẽ nhánh: Kiểm tra một vài điều kiện và rẽ nhánh chương trình.

Lệnh lặp: Thực hiện các thao tác lặp trên một số câu lệnh.

Biên dịch và thông dịch

Ngôn ngữ lập trình dùng trong bài viết này là C++ được gọi là ngôn ngữ lập trình bậc cao. Ngoài ra, bạn có thể tìm hiểu thêm các ngôn ngữ bậc cao khác như Python, Java, Ruby, JavaScript.

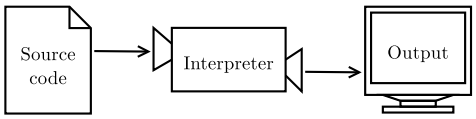
Một chương trình viết bằng ngôn ngữ lập trình bậc cao muốn chạy được trên máy tính nó phải cần dịch sang ngôn ngữ bậc thấp và được gọi là ngôn ngữ máy, quá trình này xem như là một bất lợi của ngôn ngữ lập trình bậc cao, bù lại nó có các ưu điểm sau:

- Chương trình dễ viết hơn vì ngôn ngữ bậc cao gần với ngôn ngữ tự nhiên.
- Chương trình ngắn gọn, dễ đọc và có khả năng chính xác hơn.
- Có thể chạy được trên nhiều loại máy tính khác nhau mà ít cần hoặc không phải sửa đổi.

Có hai chương trình dịch một ngôn ngữ bậc cao sang ngôn ngữ máy đó là chương trình thông dịch và biên dịch

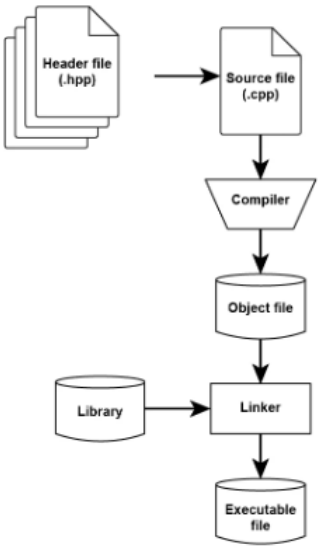
Khi một ngôn ngữ lập trình được xác định là thông dịch, chương trình được viết ra khi chạy sẽ được dịch trực tiếp thành mã máy để máy tính thực thi chúng. Khi chương trình chạy đến dòng lệnh nào sẽ chuyển thành mã máy đến đó để máy tính có thể thực thi.

Bộ thông dịch thực hiện quá trình thông dịch gọi là **interpreter**. Hình sau 1.5 sau là quy trình của một chương trình biên dịch.



Hình 1.5: Quy trình thực thi của ngôn ngữ thông dịch

Ngược lại đối với ngôn ngữ biên dịch, chương trình sẽ dịch toàn bộ thành mã máy rồi mới tiến hành thực thi. Bộ biên dịch thực hiện quá trình biên dịch được gọi là **compiler**. Hình sau 1.6 mô tả quá trình biên dịch và liên kết một chương trình viết bằng C/C++.



Hình 1.6: Quy trình thực thi của ngôn ngữ C++

Ưu điểm:

- Chương trình sau đó được thực thi nhanh hơn.
- Độ tin cậy cao.
- Khó bị dịch ngược mã nguồn.

1.1.4 Cấu trúc một bài toán

Các bài toán trong cuốn sách này có cấu trúc gồm:

Yêu cầu: là phần chúng ta nên tập trung quan tâm nhiều nhất vì thông tin trình bày ở đây chính là cái chúng ta cần giải quyết.

Dữ liệu/Kết quả: là phần rất quan trọng không kém, chúng cho ta biết dữ liệu vào được bố trí như thế nào và phải xuất ra kết quả ra sao mới đúng. Và thêm nữa đó là giới hạn, ràng buộc về dữ liệu hoặc mô tả các subtask của dữ liệu vào để ta biết cần phải giải quyết bằng cách nào là hợp lý.

Ví dụ mẫu: đây sẽ là phần ta hay nghiên cứu nhất trên đề bài. Nó là minh họa cho bài toán, cho yêu

cầu đặt ra và được xem như các trường hợp riêng. Cho nên hiểu rõ các ví dụ cũng như giải thích được chúng là một lợi thế rất lớn để có thể tổng quát hóa cho bài toán.

Ràng buộc: là miêu tả về thời gian thực hiện của thuật toán và giới hạn bộ nhớ chương trình. Lỗi TLE là lỗi mà ta cần quan sát mục này kỹ nhất.

Ngoài các phần cơ bản trên, các bài toán thường có các thông tin đính kèm cũng vô cùng quan trọng như: thể loại của bài toán, thời gian chạy và kích thước giới hạn bộ nhớ của chương trình. Trong hầu hết với các bài toán của cuốn sách này thời gian chạy được thiết lập là 1 giây (1000 ms) tương ứng với khoảng 10^7 phép tính trên một giây.

1.1.5 Ngôn ngữ lập trình C/C++

Có rất nhiều ngôn ngữ được sử dụng để học lập trình như C/C++, Java, Pascal. Ngôn ngữ C/C++ với nhiều ưu thế gồm:

- Chạy nhanh.
- Tính chia sẻ cao khi mã nguồn, tài liệu biểu diễn bằng C++ chiếm số lượng lớn.
- Thư viện hỗ trợ mạnh mẽ (STL).
- Được xem là ngôn ngữ máy tính thông dụng nhất cho hầu hết các lập trình viên.

1.1.6 Bộ ký tự, từ khóa, tên của C/C++

Bộ ký tự

Hầu hết các ngôn ngữ lập trình bậc cao C++ sử dụng bộ ký tự chuẩn của ASCII để xây dựng lên ngôn ngữ của mình, bao gồm:

- Các ký Alphabet ('a' ->'z'; 'A' ->'Z').
- Các ký tự số (0..9).
- Các ký tự đặt biệt (+, -, *, /, %, <, >, =, !, ...)

Từ (word):

Là sự kết hợp của các ký tự, các từ phân biệt nhau bởi dấu cách (space). Từ là đơn vị cơ bản của ngôn ngữ, từ đó ta có: từ khóa (keyword), tên (identifier) cho biến số, hằng số, hàm số, lớp, đối tượng sử dụng trong chương trình viết bằng C++.

Từ khóa (key word):

Là tên được định nghĩa trước bởi C++, mỗi từ có một ý nghĩa nhất định khi sử dụng trong lập trình. Sau đây là một vài từ khóa thông dụng:

Tên (identifier):

Là một từ mà người lập trình đặt cho cho biến số, hằng số, hàm số, lớp, đối tượng sử dụng trong chương trình. Theo đó, ta phải sử dụng quy tắc đặt tên cho các đối tượng nêu trên. Và hiện nay có rất nhiều quy tắc được đưa ra nhưng cơ bản vẫn là:

asm	double	new	switch
auto	else	operator	template
break	enum	private	this
case	extern	protected	throw
catch	float	public	try
char	for	register	typedef
class	friend	return	union
const	goto	short	unsigned
continue	if	signed	virtual
default	inline	sizeof	void
delete	int	static	volatile
do	long	struct	while

Bảng 1.1: Danh sách các từ khóa của c++

- Đặt tên có ý nghĩa, có thể tận dụng ký hiệu toán học để đặt.
- Không bắt đầu bằng ký tự số.
- Phải là một từ, nghĩa là không sử dụng ký tự trắng và một số ký tự đặc biệt như: +, -, *, /,...
- Bạn có thể thao khảo quy tắc Camel, tức là ghép các từ lại và viết Hoa ký tự đầu tiên của mỗi từ trừ từ đầu tiên, ví dụ: dienTichHinhTron, chieuCao, tocDo, ngaySinh.
- Phân biệt chữ HOA khác với chữ thường.
- Không sử dụng từ khóa, tên chuẩn (tên các hàm thư viện chuẩn).

Tên chuẩn:

Là từ được C++ đăng ký thông qua các thư viện chuẩn bao gồm tên các hàm, biến chuẩn. Thường để nắm bắt nhanh cấu trúc của một chương trình cũng như một số thành phần cơ bản. Chương trình in dòng chữ "Hello world!" được giới thiệu sau đây giúp bạn tìm hiểu nhanh các kiến thức vừa trình bày.

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 int main()
4 {
5    string yourName;
6    getline(cin, yourName);
7    cout<<"Hello " << yourName;
8    return 0;
9 }
```

Dòng lệnh đầu tiên hướng dẫn ta khai báo thư viện chuẩn STL của C++, thư viện này bao gói hết các thư viện quen thuộc dùng trong C++ như <iostream>, <vector>, <algorithm> v.v.

Dòng thứ hai khai báo không gian tên cho thư viện STL.

Dòng thứ năm khai báo một biến kiểu xâu ký tự.

Dòng thứ sáu nhập dữ liệu kiểu xâu từ bàn phím.

Dòng thứ bảy in dòng chữ "Hello "+ yourName

Một chương trình viết bằng C++ luôn phải có một hàm main() với giá trị trả về là 0 báo cho hệ điều hành biết mình thực hiện thành công.

Mẫu của một chương trình C++

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    //Mã lệnh của bạn bắt đầu từ đây
    return 0;
}
```

Dòng lệnh đầu tiên hướng dẫn ta khai báo thư viện chuẩn STL của C++, thư viện này bao gói hết các thư viện quen thuộc dùng trong C++ như <iostream>, <vector>, <algorithm>,v.v. Dòng thứ hai khai báo không gian tên cho thư viện STL. Một chương trình viết bằng C++ luôn phải có một hàm main() với giá trị trả về là 0 báo cho hệ điều hành biết mình thực hiện thành công.

1.2 Bài tập chương 1

Bài toán 1.1. Diện tích hình tròn

Vẽ sơ đồ khối cho thuật toán tính diện tích của một hình tròn với bán kính $r \in R$ cho trước.

Bài toán 1.2. Diện tích tam giác

Vẽ sơ đồ khối cho thuật toán tính diện tích của một tam giác với ba cạnh $a, b, c \in R$ cho trước.

Bài toán 1.3. Phương trình bậc hai

Vẽ sơ đồ khối cho thuật toán giải phương trình $ax^2 + bx + c = 0$ với $a \neq 0$ và $a, b, c \in R$ cho trước.

Bài toán 1.4. Tổng dãy số

Vẽ sơ đồ khối cho thuật toán tính tổng $S = a_1 + a_2 + \ldots + a_n; ai \in N$ với n số nguyên dương cho trước.

Bài toán 1.5. Phần tử lớn nhất

Vẽ sơ đồ khối cho thuật toán tìm phần tử lớn nhất trong một dãy $A = \{a_1, a_2, \ldots, a_n\}$ gồm n số nguyên cho trước.

Bài toán 1.6. Tìm phần tử x

Vẽ sơ đồ khối cho thuật toán kiểm tra xem phần tử x có xuất hiện trong dãy $A = \{a_1, a_2, \ldots, a_n\}$ gồm n số nguyên cho trước.

Bài toán 1.7. Tổng các chữ số

Vẽ sơ đồ khối cho thuật toán tính tổng các chữ số của một số nguyên n cho trước.

Bài toán 1.8. Đổi hệ cơ số 2

Vẽ sơ đồ khối cho thuật toán đổi một số hệ thập phân sang số hệ nhị phân.

Bài toán 1.9. Hello World

Cài đặt và sử dụng phần mềm DEV C++ (https://sourceforge.net/projects/orwelldevcpp/) để viết và chạy thử nghiệm chương trình sau:

```
1  //----------------------------
2  //author: Le Minh Triet
3  //file: D:\Programming\Hello.cpp
4  //date:12/9/2023
5  //----------------------------
6  #include<iostream>
7  using namespace std;
8  int main()
9  {
10   cout << "Hello World!";
11   return 0;
12 }
```

Bài toán 1.10. Hello Again

Cài đặt và sử dụng phần mềm Code::Blocks (https://www.codeblocks.org/downloads/) để viết và chạy thử nghiệm chương trình sau:

```
//-----------------------------
//author: Le Minh Triet
//file: D:\Programming\aplusb.cpp
//date:12/9/2023
//-----------------------------
#include<iostream>
using namespace std;
int main()
{
   //Khai báo biến số
   int a, b;
   //Nhập dữ liệu
   cin >> a >> b;
   //In kết quả
   cout << a + b << endl;
   return 0;
}
//5 6
//11
```

Chương 2

Các thành phần cơ bản của C/C++

2.1 Tóm tắt lý thuyết

2.1.1 Các kiểu dữ liệu cơ bản

2.1.1.1 Số nguyên

Số nguyên được sử dụng trong hầu hết các bài toán trong cuốn sách này và theo đó hai kiểu dữ liệu số nguyên thông dụng được sử dụng là int và long long. Trên quan điểm sử dụng cấu trúc dữ liệu, ta quan tâm đến:

- Tên kiểu: int, long long, unsigned int, unsigned long long,...
- Kích thước lưu trữ (có thể dùng sizeof(int) để tính) nhưng có thể áng chừng số nguyên int với 9 chữ số, long long với 19 chữ số. Hoặc bạn có thể dùng các hằng số SHRT_MIN, INT_MAX, ULLONG_MAX để xác định số lớn nhất, nhỏ nhất.
- Các phép toán trên số nguyên, gồm: số học (+, -, *, /, \%), quan hệ (>, <, >=, <=, ==, !=), các phép xử lý bit.
- Các hàm số học hỗ trợ như: abs(), acos(), sqr(), sqrt(), ceil(), floor(),....

Xét một số ví dụ về số nguyên như sau:

Ví dụ 2.1. Các phép toán số học

Cho hai số nguyên X và Y, lập trình in ra tổng, tích và hiệu của chúng.

Input: Dòng duy nhất chứa hai số x, y thỏa $|x, y| \leq 10^4$.

Output: In ra ba dòng gồm:

- dòng 1 in ra tổng hai số theo mẫu: x + y = summation

- dòng 2 in ra tích hai số theo mẫu: x * y = multiplication

- dòng 3 in ra hiệu hai số theo mẫu: x - y = subtraction

Sample Input 1

5 10

Sample Output 1

5 + 10 = 15

5 * 10 = 50

5 - 10 = -5

Lời giải:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
   int x, y;
   cin >> x >> y;
   cout << x << " + " << y << " = "<< x + y <<endl;
   cout << x << " * " << y << " = "<< x * y <<endl;
   cout << x << " - " << y << " = "<< x - y <<endl;
   return 0;
}
```

Ví dụ 2.2. Diện tích hình chữ nhật

Viết chương trình tính diện tích hình chữ nhật với hai cạnh a, b cho trước.

Input: Dòng duy nhất chứa hai số nguyên a, b thỏa $1 \leq a, b \leq 10^9$.

Output: In ra diện tích cần tính.

Sample Input 1

5 3

Sample Output 1

15

Lời giải:

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
int main()
{
   int a, b; cin >> a >> b;
   ll S = (ll)a*b;
   cout << S;
```

```
9     return 0;
10 }
```

Nhận xét:

Đây là một bài toán đơn giản, lỗi duy nhất có thể xảy ra đối với người học ở câu lệnh long long S= a*b; với thiết kế ban đầu theo yêu cầu bài toán là cạnh chữ nhật khai báo kiểu int a, b. Phép toán này không đủ để chứa dữ liệu nếu số lớn (a=123456789, kết quả S=-1757895751). Ở bài toán này ta dùng một kỹ thuật gọi là ép kiểu (câu lệnh dòng số 7).

Điểm chú ý của kiểu số nguyên chính là phép toán chia (/), đây là phép chia nguyên cho nên 5/2 cho kết quả là 2.

Phép toán đặc biệt nhất của số nguyên chính là phép toán modulo (%). Phép toán này được áp dụng cho tính chất **chia hết** và với hệ quả của nó ta dùng để xét tính chất **chẵn, lẻ** của một số nguyên. Ngoài ra nó còn dùng cho trường hợp xử lý số nguyên lớn, kiểu như một vành số nguyên và các con số tính toán chỉ thuộc phạm vi của số cần modulo (M).

Số thường được chọn là số $M = 10^9 + 7$ đây là một số nguyên tố có giá trị gần lớn nhất trong kiểu số nguyên int. Ngoài ra việc chọn nó là một số nguyên tố giúp hỗ trợ thêm cho phép chia (được xét dưới dạng nghịch đảo). Ba phép toán chính được dùng với quy tắc modulo là:

$$(a+b)\%M = (a\%M + b\%M)\%M$$
$$(a-b)\%M = (a\%M - b\%M)\%M$$
$$(a*b)\%M = (a\%M * b\%M)\%M$$

Ứng dụng của phép modulo sẽ được tìm hiểu kỹ trong phần lý thuyết số.

Ví dụ 2.3. Tổng và tích

Cho hai số nguyên $m = \overline{abcd}$ và $n = \overline{efgh}$. Viết chương trình tính $S = d + h, P = a \times e$.

Input: Dòng duy nhất chứa hai số nguyên m, n thỏa $1000 \leq m, n \leq 9999$.

Output: In ra kết quả cần tính cho S, P, mỗi số in trên một dòng.

Sample Input 1

2345 9876

Sample Output 1

11

18

Lời giải:

```
1 #include <bits/stdc++.h>
2 using namespace std;
3 int main()
```

```
{
   int m, n; cin >> m >> n;
   S = m % 10 + n % 10;
   P = m/1000 * n/1000;
   cout << S << endl << P <<endl;
   return 0;
}
```

2.1.1.2 Số thực

Hoàn toàn tương tự như số nguyên, ta xem xét số thực với các tính chất sau:

- Tên kiểu (float, double, long double).
- Kích thước lưu trữ (có thể dùng sizeof(double) để tính).
- Miền dữ liệu ta có thể dùng hàm sau để tính: numeric_limits<long double>::max(); hoặc numeric_limits<long double>::mi
- Các phép toán trên số thực gồm: số học (+, -, *, /), quan hệ (>, <, >=, <=, ==, !=).
- Các hàm số học hỗ trợ như:
  abs(), acos(), sqr(), sqrt(), ceil(), floor(),....

Vậy điểm khác biệt giữa số nguyên và số thực dễ nhận biết chính là tính chất đếm được (nguyên) và không đếm được (hơn kém nhau một espilon). Cho nên kiểu số nguyên thì có phép modulo còn kiểu số thực thì không.

Ví dụ 2.4. Hàm ceil, floor và round

Viết chương trình tính các giá trị ceil, floor, round của a/b với hai số nguyên dương a, b cho trước.

Input: Dòng duy nhất chứa hai nguyên dương a, b thỏa $1 \leq a, b \leq 10^3$.

Output: In ra các giá trị cần tính, gồm:

- dòng thứ nhất in ra giá trị hàm floor.

- dòng thứ hai in ra giá trị hàm ceil.

- dòng thứ ba in ra giá trị hàm round.

Sample Input 1

10 3

Sample Output 1

3

4

3

Lời giải:

```
#include<bits/stdc++.h>
```

```
using namespace std;
int main()
{
   int a, b; cin >> a >> b;
   cout << floor((double)a/b)<<endl;
   cout << ceil((double)a/b)<<endl;
   cout << round((double)a/b)<<endl;
   return 0;
}
```

Thông thường số chấm động kiểu float có độ chính xác đơn (single-precision) chính xác đến 7 chữ số. Kiểu double có độ chính xác kép (double-precision) chính xác đến 16 chữ số. Cho nên dẫn đến sai lệch số khi tính toán với số thực và phép so sánh trên số thực cần thực hiện một cách cẩn thận khi xem xét đến độ chính xác của nó, xét ví dụ sau:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
   cout<< fixed << showpoint << setprecision(17);
   double a = 1.0;
   double b = 3.0/7.0 + 2.0/7.0 + 2.0/7.0;
   cout << "r = "<< (a == b) << endl;
   cout << "a = "<< a << endl;
   cout << "b = "<< b << endl;
   return 0;
}
```

Các kết quả in ra khi chạy chương trình:

r = 0

a = 1.00000000000000000

b = 0.99999999999999989

Nếu xem xét theo kiểu toán học thì a và b có giá trị bằng nhau, tuy nhiên khi thực hiện các phép toán cộng và chia trên số thực như trên thì a và b không còn bằng nhau nữa. Vậy đây là một ví dụ mà bạn cần phải chú ý khi lập trình với số thực.

Bạn nên sử dụng kiểu double khi cần lưu trữ một số chấm động, hạn chế sử dụng float vì kiểu float có độ chính xác thấp sẽ dẫn tới số không chính xác.

Ví dụ 2.5. Diện tích hình tròn

Viết chương trình tính diện tích hình tròn với bán kính r cho trước.

Input: Dòng duy nhất chứa số thực r.

Output: In ra diện tích cần tính với 6 chữ số sau dấu chấm thập phân.

Sample Input 1

2

Sample Output 1

12.566371

Chú ý: Cho trước giá trị của số pi là const double pi=acos(-1);

Lời giải:

```
#include <bits/stdc++.h>
using namespace std;
const double pi=acos(-1);
int main()
{
   cout << fixed << setprecision(6);
   double r, S=0; cin >> r;
   S = pi * r * r;
   cout << S;
   return 0;
}
```

Nhận xét:

Với đề bài chỉ đề cập đến số thực mà không chỉ rõ kiểu dữ liệu với thêm yêu cầu độ chính xác của kết quả lên đến 6 chữ số thập phân cho nên tốt hơn hết là dùng kiểu double. Nếu dùng kiểu float sự chênh lệch của kết quả là có thể xảy ra.

Ví dụ với kiểu double mà $r = 123.456789$ ta có kết quả $S = 47882.831831$, nếu thay bằng kiểu float ta nhận kết quả $S = 47882.832031$.

2.1.2 Biến, hằng, hàm xây dựng sẵn

Biến (variable):

Biến (variable) đại diện cho vùng nhớ lưu trữ dữ liệu (trên RAM) của chương trình. Biến chứa giá trị nhập vào, giá trị của một biểu thức, giá trị tính toán hoặc xử lý trong chương trình. Biến được nhận dạng thông qua **tên biến** và **kiểu dữ liệu**.

Biến cần được khai báo trước khi dùng (đọc/ghi giá trị). Chương trình tự động cấp phát vùng nhớ khi một biến được khai báo.

Cú pháp khai báo biến:

<TYPE> nameVar;

hoặc

<TYPE> nameVar = values;

Trong đó:

- TYPE là kiểu dữ liệu, ví dụ số nguyên (int , long long), số thực (double), xâu ký tự (string),...
- varName tên của biến số do người lập trình đặt.
- values giá trị được gán (=) vào cho biến và phải cùng kiểu dữ liệu được khai báo.

Nếu có nhiều biến cùng kiểu dữ liệu với nhau thì có thể khai báo như sau:

<TYPE> nameVar1, nameVar2,.., nameVarn;

Phạm vi lưu trữ và truy xuất của biến

Là vùng nhớ của chương trình mà một biến tồn tại và sử dụng được. Về cơ bản chia làm hai khu vực là toàn cục và địa phương (mô tả chi tiết ở các phân đoạn nhớ ở chương sau).

Toàn cục (global): bên ngoài tất cả các hàm, được sử dụng trong toàn chương trình.

Cục bộ (local): chỉ sử dụng bên trong hàm, vị trí gồm:

- Thân hàm: từ dấu { đến dấu } của thân hàm.

- Hoặc các khối con (từ dấu { đến dấu } của khối).

Xét ví dụ đoạn mã sau:

```
1  #include <iostream>
2  using namespace std;
3  // Khai báo biến toàn cục
4  int c = 12;
5  void test(){
6     // Truy cập và thay đổi biến toàn cục
7     ++c;
8     cout << c << endl;
9  }
10 int main(){
11    // Truy cập và thay đổi biến toàn cục
12    ++c;
13    cout << c <<endl;
14    test();
15    //khai báo biến địa phương
16    int c=2;
17    cout << c <<endl;
18    return 0;
19 }
```

Chương trình trên sẽ in ra:

13

2

Hằng (constance):

Vùng nhớ có khả năng thay đổi hoặc không cho thay đổi dữ liệu. Các vùng nhớ mà giá trị dữ liệu có thể thay đổi được gọi là các biến. Các vùng nhớ mà giá trị dữ liệu được khởi tạo một lần và không bao giờ thay đổi được gọi là các hằng số. Chúng thường được định nghĩa trong vùng toàn cục và được đặt trước bởi từ khóa const hoặc thông qua cú pháp #define.

#define nameConst values

const <TYPE> nameConst = values;

ví dụ khai báo như sau: const double PI = 3.14; hoặc #define PI 3.14

Hàm dựng sẵn (build-in function):

Trong thư viện chuẩn của C++ cung cấp khá nhiều hàm dựng sẵn giúp ích cho người lập trình, và một trong số đó là các hàm toán học. Để sử dụng các hàm toán học trong C++, đầu tiên ta cần khai báo thư viện và không gian tên chứa chúng:

#include <cmath>

using namespace std;

Hoặc bạn dùng luôn gói bao hàm #include/bits/stdc++.h>

Thư viện cung cấp một số hàm thú vị như sau:

Tên hàm	Kiểu trả về	ý nghĩa
fabs(x)	số thực	trả về giá trị tuyệt đối của x
sin(x)	số thực	trả về sin của góc x tính băng radian
cos(x)	số thực	trả về cos của góc x tính băng radian
tan(x)	số thực	trả về tan của góc x tính băng radian
exp(x)	số thực	trả về e^x
log(x)	số thực	trả về logarit tự nhiên của x
log2(x)	số thực	trả về logarit cơ số 2 của x
pow(x,y)	số thực	trả về lũy thừa x^y
sqrt(x)	số thực	trả về căn bậc hai của x

Bảng 2.1: Danh sách các hàm của c++

Ngoài ra, C++ còn cung cấp một thư viện với rất nhiều hàm hữu ích đó là <algorithm.h> được chứa gộp trong thư viện <bits/stdc++.h>. Liệt kê một số hàm sau để các bạn tự tra cứu:

- Hàm sort(): sắp xếp dữ liệu.
- Hàm max(): tìm phần tử lớn nhất.
- Hàm min(): tìm phần tữ nhỏ nhất.
- Hàm reverse(): hàm đảo ngược xâu ký tự, đảo ngược mảng, vector.
- Hàm find(): tìm kiếm phần tử trong mảng, vector.

- Hàm count(): hàm đếm số phần tử trong mảng thỏa điều kiện cho trước.
- Hàm binary_search(), lower_bound(), upper_bound(): hàm tìm kiếm nhị phân với 3 dạng khác nhau.
- Hàm next_permutation(): hàm trả về hoán vị kế tiếp.

Tham khảo: https://cplusplus.com/reference/algorithm/.

2.1.3 Biểu thức và toán tử

Một biểu thức (expresion):

Là tổ hợp các **toán tử** và **toán hạng** được kết hợp với nhau theo một trình tự nhất định. Trong quá trình thực thi chương trình một biểu thức sẽ được lượng giá (tính toán) dựa vào giá trị cụ thể của các toán hạng tham gia biểu thức cùng với thứ tự thực hiện của các phép toán. Biểu thức sẽ có một giá trị duy nhất sau khi lượng giá. Vậy tránh trường hợp nhập nhằng về giá trị thì khi ta viết biểu thức sự kết hợp của toán tử và toán hạng phải theo trình tự đúng của biểu thức.

Toán tử:

- Toán tử số học (+, -, *, /, %).
- Toán tử quan hệ (<, <=, >, >=, ==, !=).
- Toán tử logic (not, and, or).
- Toán tử xử lý bít (!, &, |).
- Toán tử tăng, giảm tự động (++,--)

Toán hạng:

- Biến (variable).
- Hằng (constance)
- Hàm (function).
- Đối tượng (object).

Xét ví dụ: Viết biểu thức bằng C++ cho công thức $x_1 = \dfrac{-b + \sqrt{\Delta}}{2a}$.

Một số cách viết biểu thức C++, delta thay cho ký hiệu toán học Δ, x1 thay cho x_1, các biểu thức tương ứng có thể là:

x1 = (-b+sqrt(delta))/2/a;

x1 = (-b+sqrt(delta))/(2*a);

x1 = (-b+sqrt(delta))/2*a; Biểu thức này nhận giá trị sai khi lượng giá do thứ tự ưu tiên của phép toán.

Do có sự kết hợp của các dạng toán tử khác nhau cho nên khi viết biểu thức cần xác định thứ tự của các phép toán. Nhưng hay hơn cả vẫn là phép sử dụng dấu ngoặc (được gọi là phép toán ma thuật) nó giúp chúng ta viết chính xác phép toán cần tính, thay đổi thứ tự ưu tiên một cách tường minh. Và vẫn phải tuân theo thứ tự "Ngoặc trong tính trước, ngoặc ngoài tính sau, nếu bằng nhau tính từ trái sang".

Chú ý: Dùng mở ngoặc đóng ngoặc cho chính xác.

Thứ tự	Phép toán	Tính liên kết
1	(), [], a++, a--	Từ trái sang phải
2	!, ~, ++a, --a	Từ phải sang trái
3	*, /, \%	Từ trái sang phải
4	+, -	Từ trái sang phải
5	<<, >>	Từ trái sang phải
6	<, <=, >, >=, !=, ==	Từ trái sang phải
7	!, &, ^	Từ trái sang phải
8	not, and, or	Từ trái sang phải

Bảng 2.2: Thứ tự của các phép toán

2.1.4 Xuất, nhập dữ liệu

Đối với viết một chương trình, việc kiểm soát tiêu chuẩn đầu vào, đầu ra của dữ liệu là vô cùng quan trọng. Khi làm việc với ngôn ngữ C/C++, đầu tiên phải nói đến thư viện <stdio.h>. Thư viện cung cấp hai hàm hỗ trợ nhập xuất cơ bản là printf() và scanf(). Tuy nhiên sẽ dễ dàng hơn nếu bạn sử dụng các đối tượng xuất nhập của C++ là cin và cout.

Đây cũng là hai chức năng quan trọng của một bài toán lập trình, người lập trình phải tuân thủ nghiêm ngặt định dạng vào, ra của bài toán.

Lỗi vào ra:

Lỗi Wrong Answer thường xảy ra đối với người lập trình vì không tuân thủ định dạng vào, ra của bài toán. Một kỹ thuật đơn giản là có thể kiểm tra xem việc vào, ra có thực hiện được hay không trước khi bắt tay vào giải bài toán. Theo đó ta đọc dữ liệu vào cấu trúc dữ liệu lưu trữ ở bộ nhớ, xuất ra màn hình. Khi in kết quả thì xem kỹ định dạng mà bài toán yêu cầu, nhất là phân biệt về độ rộng, số chữ số sau dấu chấm thập phân, các ký tự văn bản trong định dạng in.

Sau khi phân tích và chọn cấu trúc dữ liệu tốt nhất cho bài toán của mình thì công việc vào và ra dữ liệu sẽ được thực hiện. Cho dù có cấu trúc dữ liệu như thế nào thì cũng chung quy lại về các đối tượng cụ thể như **số nguyên**, **số thực**, **xâu ký tự**.

Vào ra dữ liệu với C++ rất linh hoạt với hai đối tượng cin, cout. Theo thống kê hai hàm chuẩn scanf(), print() của C chạy nhanh hơn. Tuy nhiên, việc sử dụng cin, cout của thư viện iostream đơn giản hơn nhiều.

Để hình dung cho việc vào ra dữ liệu ta xét chương trình sau:

```
1 #include <iostream>
2 using namespace std;
3 int main()
4 {
5   int a, b, c;
6   cin >> a >> b >> c;
7   cout << "a = "<< a << endl;
8   cout << "b = "<< b << endl;
9   cout << "c = "<< c << endl;
```

```
10      return 0;
11  }
```

Các giá trị được nhập vào và in ra như sau:

12334 625 23

a = 12334

b = 625

c = 23

Việc in dữ liệu với số thực cần phải định dạng kích thước, ta sử dụng cout << fixed << setprecision(9); với 9 là số chữ số sau dấu chấm thập phân và bạn có thể thay đổi theo yêu cầu bài toán.

2.2 Bài tập chương 2

Bài toán 2.1. Các phép tính số học bản 1

Viết chương trình thực hiện các phép tính $+, -, *, /, \%$ trên hai số nguyên a, b.

Input: Dòng duy nhất chứa hai số a, b thỏa $0 \leq a \leq 10^4; 1 \leq b \leq 10^4$.

Output: In kết của các phép tính trên năm dòng theo mẫu:

dòng 1: a + b = result

dòng 2: a - b = result

dòng 3: a * b = result

dòng 4: a / b = result

dòng 5: a % b = result

Sample Input 1

9999 45

Sample Output 1

9999 + 45 = 10044

9999 - 45 = 9954

9999 * 45 = 449955

9999 / 45 = 222

9999 % 45 = 9

Link nộp bài: https://coder.husc.edu.vn/problem/pb0001

Bài toán 2.2. Các phép tính số học bản 2

Viết chương trình thực hiện các phép tính $+, -, *, /, \%$ trên hai số nguyên a, b.

Input: Dòng duy nhất chứa hai số a, b thỏa $-10^9 \leq a, b \leq 10^9, b \neq 0$.

Output: In kết của các phép tính trên năm dòng theo mẫu:

dòng 1: a + b = result

dòng 2: a - b = result

dòng 3: a * b = result

dòng 4: a / b = result

dòng 5: a % b = result

Sample Input 1

12345678 98765

Sample Output 1

12345678 + 98765 = 12444443

12345678 - 98765 = 12246913

12345678 * 98765 = 1219320887670

12345678 / 98765 = 125

12345678 % 98765 = 53

Link nộp bài: https://coder.husc.edu.vn/problem/pb0002

Bài toán 2.3. Hiệu của hai tích số

Viết chương trình thực hiện các phép tính $X = a \times b - c \times d$.

Input: Dòng duy nhất chứa bốn số nguyên a, b, c, d thỏa điều kiện $-10^9 \leq a, b, c, d \leq 10^9$.

Output: In ra kết quả theo mẫu: "Difference = X".

Sample Input 1

1 2 3 4

Sample Output 1

Difference = -10

Link nộp bài: https://coder.husc.edu.vn/problem/pb0003

Bài toán 2.4. Tổng và tích

Cho hai số nguyên có bốn chữ số $m = \overline{abcd}$ và $n = \overline{efgh}$. Viết chương trình tính $S = d + h, P = a \times e$.

Input: Dòng duy nhất chứa hai số nguyên có bốn chữ số m, n.

Output: Dòng thứ nhất in số S, dòng thứ hai in số P.

Sample Input 1

2345 9876

Sample Output 1

11

18

Link nộp bài: https://coder.husc.edu.vn/problem/pb0004

Bài toán 2.5. Đếm số chia hết cho 3

Viết chương trình đếm xem từ 1 đến n có bao nhiêu số chia hết cho 3.

Input: Dòng duy nhất chứa số nguyên n thỏa $1 \leq n \leq 10^{18}$.

Output: In kết quả cần đếm.

Sample Input 1

100

Sample Output 1

33

Link nộp bài: https://coder.husc.edu.vn/problem/pb0005

Bài toán 2.6. Các hàm toán học bản 1

Viết chương trình tính giá trị hàm số sau $f(x) = sin(x) + \sqrt{log_4 3x} + \lceil 3e^x \rceil$.

Input: Dòng duy nhất chứa thực x thỏa $1 \leq x \leq 100$.

Output: In ra kết quả của hàm với 6 chữ số sau dấu chấm thập phân.

Sample Input 1

2.3

Sample Output 1

31.926086

Link nộp bài: https://coder.husc.edu.vn/problem/pb0006

Bài toán 2.7. Các hàm toán học bản 2

Viết chương trình tính giá trị hàm số sau: $f(x) = lg^3(a) + cos^5(x)$, trong đó $a = c^4 + k^3, c = \sqrt{|x|}$.

Input: Dòng duy nhất chứa hai số thực x, k thỏa $1 \leq x, k \leq 100$.

Output: In ra kết quả của hàm với 2 chữ số sau dấu chấm thập phân.

Sample Input 1

5 8.2

Sample Output 1

21.04

Link nộp bài: https://coder.husc.edu.vn/problem/pb0007

Bài toán 2.8. Diện tích hình tròn

Viết chương trình tính diện tích hình tròn với bán kính r cho trước. Số pi được cho là 3.141592653.

Input: Dòng duy nhất chứa số thực r.

Output: In ra diện tích cần tính với 9 chữ số sau dấu chấm thập phân.

Sample Input 1

45.62514162

Sample Output 1

6539.707492001

Link nộp bài: https://coder.husc.edu.vn/problem/pb0008

Bài toán 2.9. Khoảng cách Euclide

Viết chương trình tính khoảng cách giữa hai điểm phân biệt $A(x_1, y_1)$ và $B(x_2, y_2)$ trong mặt phẳng tọa độ Oxy.

Input: Dòng duy nhất chứa bốn số nguyên x_1, y_1, x_2, y_2 thỏa $-10^9 \leq x_1, y_1, x_2, y_2 \leq 10^9$.

Output: In ra khoảng cách cần tính làm tròn đến 10^{-9}.

Sample Input 1

0 0 1 1

Sample Output 1

1.414213562

Link nộp bài: https://coder.husc.edu.vn/problem/pb0009

Bài toán 2.10. Diện tích tam giác theo góc

Viết chương trình tính diện tích của một tam giác với dữ liệu là hai cạnh a, b và góc α của hai cạnh đó.

Input: Dòng duy nhất chứa ba số thực a, b , α tính theo độ. Số pi được khai báo như sau: $pi = 3.14159$;

Output: In ra diện tích cần tính với hai chữ số sau dấu chấm thập phân.

Sample Input 1

7 10 25

Sample Output 1

14.79

Link nộp bài: https://coder.husc.edu.vn/problem/pb0010

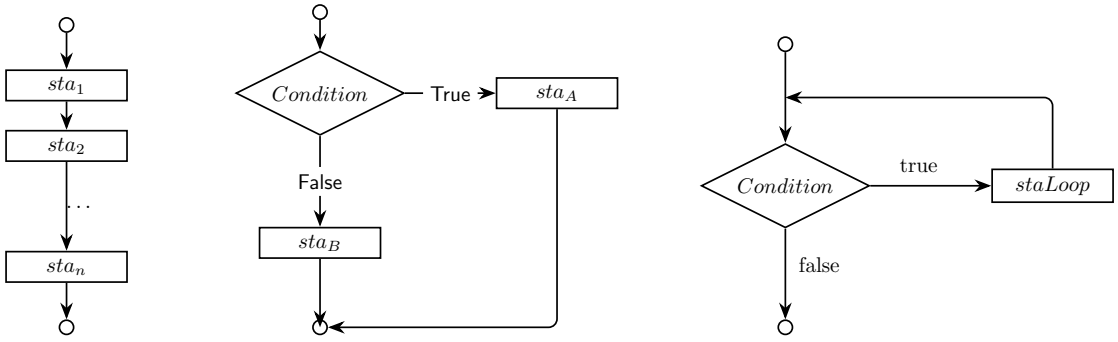
Chương 3

Cấu trúc lệnh điều khiển

3.1 Tóm tắt lý thuyết

Một chương trình máy tính có thể thực hiện bằng một trong các cách sau: Theo thứ tự tuần tự (sequence), rẽ nhánh phụ thuộc vào điều kiện, hoặc thực hiện câu lệnh lặp dựa vào điều kiện, hoặc gọi hàm. Trong đó lệnh rẽ nhánh, lệnh lặp được gọi là các cấu trúc lệnh điều khiển.

Ngoài dạng thứ nhất với cấu trúc tuần tự được xem là một chương trình đơn giản, các dạng còn lại được bố trí lệnh theo thuật toán cần giải cho bài toán và rõ ràng nếu ta thiết kế thuật toán sai thì chắc chắn sẽ trả về kết quả sai, hoặc thuật toán kém hiệu quả (theo tiêu chí thời gian). Cho nên, việc rèn luyện viết chương trình với các cấu trúc lệnh điều khiển sẽ là công việc khó khăn, cần đầu tư nhiều thời gian đối với người học.



(a) Lệnh tuần tự (b) Lệnh rẽ nhánh (c) Lệnh lặp

Hình 3.1: Các dạng lệnh của chương trình.

3.1.1 Biểu thức điều kiện

Các dạng lệnh điều khiển rẽ nhánh, lặp đều dựa trên một công thức gọi là điều kiện để thực hiện lệnh, công thức này còn được gọi là biểu thức logic, nghĩa là khi ước lượng (tính toán) giá trị của nó sẽ trả về

một trong hai giá trị là đúng true hoặc sai false. Để xây dựng được biểu thức logic này, người ta dựa vào các phép toán sau:

- Phép toán so sánh: (<, <=, >, >=, ==, !=).
- Phép toán logic: not, and, or hoặc viết bằng !, &&, | |
- Và sự kết hợp của các dạng phép toán: số học, so sánh, logic, xử lý bit.
- C++ bổ sung thêm lệnh so sánh ba ngôi cho biểu thức điều kiện.

Các phép toán quan hệ

Phép toán	Ý nghĩa	Sử dụng
<	nhỏ hơn	a < b
<=	nhỏ hơn hoặc bằng	a <= b
>	lớn hơn	a > b
>=	lớn hơn hoặc bằng	a >= b
==	bằng nhau	a == b
!=	khác nhau	a != b

Bảng 3.1: Các phép toán quan hệ

Phép so sánh trên số nguyên int rất trong sáng, rõ ràng còn trên số thực double, float cực kỳ phức tạp lý do khi tính toán với số thực ta có giá trị gần đúng (epsilon), cho nên hai số thực khi so sánh cần tính đến trường hợp này.

Khi thực hiện các phép toán quan hệ trên, kết quả trả về sẽ có giá trị logic (bool) bao gồm một trong hai giá trị là true và false.

Trong các phép toán trên, phép so sánh bằng (==) là phép toán khá phức tạp khi so sánh với dữ liệu số thực vì số thực khi tính toán sẽ xuất hiện sai số. Bạn nên cẩn thận khi so sánh bằng nhau giữa các số thực. Một cách để kiểm tra xem hai số thực có bằng nhau hay không là kiểm tra xem giá trị tuyệt đối của hiệu số của chúng có nhỏ hơn một số rất nhỏ (epsilon) cho trước hay không?. Giả sử x và y là các số dấu phẩy động và dung sai là esp=0.00000001. Biểu thức fabs (x-y) <esp xác định x có bằng y hay không. Ngoài ra người lập trình còn dễ nhầm lẫn với phép toán gán bằng (=).

Các phép toán Logic

Gồm ba phép toán cơ bản: not, and, or. Sự kết hợp của các phép toán số học, phép so sánh và phép logic

A	B	$not(A)$	$A\ and\ B$	$A\ or\ B$
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

Bảng 3.2: Các phép toán logic

tạo ra các điều kiện để ta thực thi các luồng câu lệnh trong chương trình. Và do có sự kết hợp cho nên khi viết biểu thức logic cần xác định thứ tự của các phép toán.

3.1.2 Lệnh đơn, khối lệnh

Lệnh đơn giản (simple statements) bao gồm các câu lệnh sau:

- Lệnh gán, ví dụ int a=5;
- Lệnh nhập dữ liệu, ví dụ cin >> a >> b >> c;
- Lệnh in dữ liệu, ví dụ cout<< sqrt(x);
- Các lệnh khai báo biến, hằng, hàm số.
  Ví dụ int a, b, c; const double pi=3.14; void solve(int n);
- Các lệnh khai báo, định nghĩa kiểu dữ liệu.
  Ví dụ typedef long long ll; typedef struct Ali{ double x, double y} Point;

Ngoài ra, hầu như hết trong các ngôn ngữ lập trình bậc cao để mô tả sự logic của thuật toán, khi cần kết hợp của hai hay nhiều lệnh để cần thực hiện một công việc người ta sẽ nhóm chúng lại với nhau và gọi là khối lệnh. C++ dùng cặp ký tự { block_statements }, ví dụ đoạn mã sau:

```
1   if (delta>0){
2      x1= (-b + sqrt(delta))/(2*a);
3      x2= (-b - sqrt(delta))/(2*a);
4   }
```

Khối lệnh thường được sử dụng trong các cấu trúc lệnh điều khiển.

3.1.3 Lệnh rẽ nhánh

Có những tình huống xảy ra trong cuộc sống thực khi chúng ta cần phải đưa ra một số quyết định và dựa trên những quyết định này, chúng ta quyết định mình nên làm gì tiếp theo. Các tình huống tương tự cũng nảy sinh trong lập trình, nơi chúng ta cần đưa ra một số quyết định và dựa trên những quyết định này, chúng ta sẽ thực thi khối mã tiếp theo. Lệnh rẽ nhánh if..else của C++ thực hiện công việc chọn luồng thực thi các câu lệnh dựa vào một biểu thức logic cho trước. Các dạng sử dụng lệnh if..else của C++ thể hiện như sau:

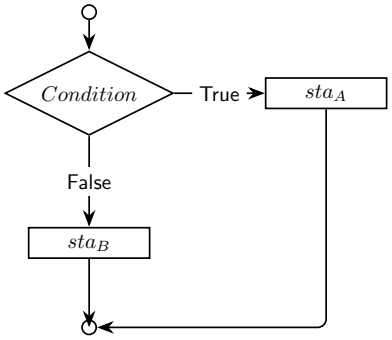
Lệnh if dạng đầy đủ

Dạng đầy đủ của lệnh if có hai nhánh như sơ đồ khối như hình vẽ sau:

Cú pháp:

```
1  if (Condition==true){
2     StaA;
3  }
4  else{
5     StaB;
6  }
```

Hoạt động: Nếu biểu thức logic (Condition) có giá trị đúng (true) thì nhóm câu lệnh A (Sta_A) được thực hiện, **ngược lại** nhóm câu lệnh B (Sta_B) được thực hiện. Như vậy chỉ một trong hai nhóm lệnh được thực hiện tùy vào giá trị của biểu thức điều kiện.



Hình 3.2: Sơ đồ khối lệnh if..else.

Ví dụ 3.1. Đếm số toa tàu

Một tàu hỏa cần chở n khách tham quan. Biết rằng mỗi toa có p khoang, mỗi khoang có 4 chỗ ngồi. Hỏi cần mấy toa để chở hết khách tham quan.

Input: Dòng duy nhất chứa hai số nguyên n, p thỏa $1 \leq n \leq 10^5; 1 \leq p \leq 2.10^4$.

Output: In ra kết quả cần tính.

Sample Input 1

892 10

Sample Output 1

23

Sample Input 2

234 10

Sample Output 2

6

Link nộp bài: https://coder.husc.edu.vn/problem/pb0018

Lời giải:

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
   int n, p;
   cin >> n >> p;
   //Tính số chổ ngồi của một toa.
   int c = p * 4;
```

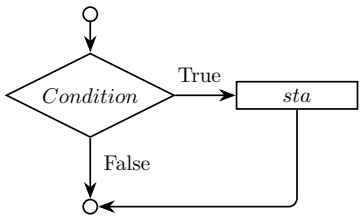
```
10    //Nếu số hành khách chia hết số chổ ngồi của một toa thì kết quả sẽ là n/c ngược lại thì phải thêm
   1 toa cho số người dư ra.
11    if (n%c ==0) cout << n/c;
12    else cout<< (n/c) + 1;
13    return 0;
14  }
```

Bạn có thể hoán đổi sự thực hiện của hai nhóm lệnh bằng cách phủ định biểu thức logic (!Condition).

Chú ý: Biểu thức điều kiện của lệnh if phải đặt trong cặp dấu ngoặc (). Biểu thức (a%b==0 or b%a==0) được viết lại một cách tường minh bằng cách dùng dấu ngoặc như sau ((a%b==0) or (b%a==0)), khi đó ta không cần phải nhớ độ ưu tiên của phép toán or với phép so sánh bằng ==.

Lệnh if dạng khuyết

Dạng khuyết của lệnh if có một nhánh được thực hiện nếu thỏa điều kiện và có sơ đồ khối như hình vẽ sau:



Hình 3.3: Sơ đồ khối lệnh if..

Cú pháp:

```
1  if (Condition==true){
2     Sta;
3  }
```

Hoạt động: Nếu biểu thức logic (Condition) có giá trị đúng (true) thì nhóm câu lệnh A (*sta*) được thực hiện và thoát ra cấu trúc if.

Ví dụ 3.2. Chính xác một điều kiện

Với số nguyên n cho trước, hãy viết chương trình in ra Yes nếu thỏa chính xác một trong các điều kiện sau và No nếu ngược lại.

- Nếu n là số chẵn.

- Nếu n nhỏ hơn 0 và chia hết cho 3.

Input: Dòng duy nhất chứa số nguyên n thỏa $|n| \leq 10^9$.

Output: In ra Yes nếu thỏa và No nếu ngược lại.

Sample Input 1

-6

Sample Output 1

No

Link nộp bài: https://coder.husc.edu.vn/problem/pb0022

Lời giải:

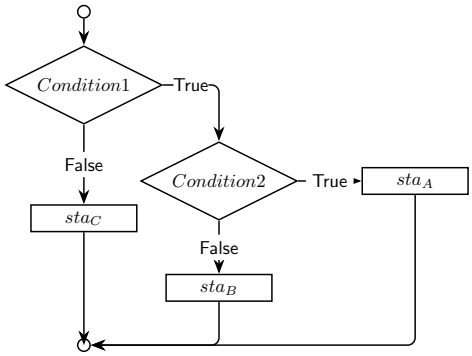
Bài toán này có thể giải bằng cách suy nghĩ đơn giản như sau: Đếm xem số n thỏa mãn bao nhiêu điều kiện nêu ra, nếu đúng chính xác một điều kiện thì in **Yes** và **No** nếu ngược lại.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
   int n; cin >> n;
   int cnt = 0;
   if (!(abs(n) % 2))   cnt++;
   if (n < 0 && !(abs(n) % 3)) cnt++;
   cout << (cnt == 1 ? "Yes" : "No");
   return 0;
}
```

Lệnh if dạng lồng nhau

Dạng lồng nhau của lệnh if có sơ đồ khối như hình vẽ sau:



Hình 3.4: Sơ đồ khối lệnh if..else lồng nhau

Cú pháp:

```
   if (Condition1 == true){
      if (Condition2 == true){
         StaA;
```

```
    }
    else{
      StaB;
    }
  }
  else{
    StaC;
  }
```

Về cơ bản nguyên tắc hoạt động giống câu lệnh if..else, ta đồng nhất nhóm lệnh A (sta_A) với một câu lệnh if..else mà điều kiện của nó là (Condition2). Xét ví dụ bài toán sau:

Ví dụ 3.3. Giải phương trình bậc nhất

Viết chương trình tìm giá trị của x trong phương trình $ax + b = 0$.

Input: Dòng duy nhất chứa hai số nguyên a, b thỏa $|a, b| \leq 10^9$.

Output: In ra kết quả cần tìm với hai số thập phân sau dấu phẩy động. Nếu không tìm thấy kết quả thì in một trong các thông điệp là **No Solution** với trường hợp không tính được hoặc **Many Solutions** với trường hợp vô số nghiệm.

Sample Input 1

5 -7

Sample Output 1

1.40

Sample Input 2

0 2

Sample Output 2

No Solution

Link nộp bài: https://coder.husc.edu.vn/problem/pb0014

Lời giải:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
  int a, b; cin >> a >> b;
  if (a==0){
    if (b==0) cout << "Many Solutions\n";
    else cout << "No Solution\n";
  }
```

```
10    else {
11        double x = (double)(-b) / a;
12        cout << fixed << setprecision(2) << x << endl;
13    }
14    return 0;
15 }
```

3.1.4 Lệnh lặp for

Các cấu trúc lặp trong lập trình được chia thành hai dạng: Lặp với số lần lặp xác định và lặp với số lần lặp không xác định. C++ có hai cấu trúc lệnh cho hai dạng trên là for cho vòng lặp với số lần lặp xác định và while với điều kiện đầu cho vòng lặp với số lần lặp không xác định.

Xét bài toán sau minh họa cho lệnh lặp xác định.

Ví dụ 3.4. In 5 dòng thông điệp "Hello world!"

Viết chương trình in 5 dòng thông điệp "Hello world!" lên màn hình.

Input: Không có

Output In ra theo yêu cầu, mỗi thông điệp in trên một dòng.

Sample Output

Hello world!

Hello world!

Hello world!

Hello world!

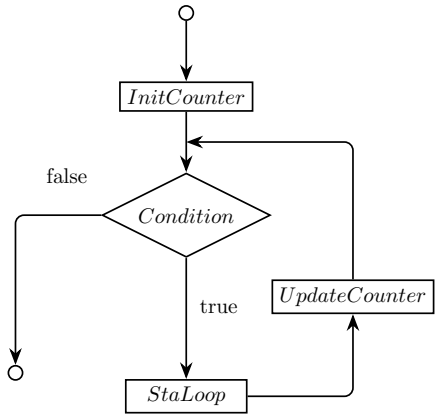
Hello world!

Lời giải.

Bài này được giải một cách đơn giản là viết câu lệnh in thông báo năm lần, vấn đề là năm câu lệnh trên đều giống nhau hoàn toàn nên trong lập trình ta gọi là câu lệnh lặp (loop) và số lần lặp lại xác định. Hầu hết các ngôn ngữ lập trình đều hỗ trợ cấu trúc cho việc điều khiển lệnh lặp này, cụ thể C++ có thể viết lại bài toán trên như sau:

```
1 #include <bits/stdc++.h>
2 using namespace std;
3
4 int main() {
5    for (int c=1; c<=5; c++)
6        cout << "Hello World!";
7    return 0;
8 }
```

Sơ đồ khối cho cấu trúc lệnh for của hầu hết các ngôn ngữ lập trình bậc cao như sau:



Hình 3.5: Sơ đồ khối lệnh for

Đây là sơ đồ khối chuẩn cho vòng lặp for đối với hầu hết các ngôn ngữ, trong đó nó sử dụng một bộ đếm với ba thao tác chính:

- Khởi tạo **giá trị đầu** cho bộ đếm, thường là giá trị 1 (như ví dụ trên c = 1).

- So sánh giá trị của bộ đếm xem đã đạt **giá trị cuối** cùng hay chưa ở đây thể hiện ở biểu thức điều kiện (như ví dụ trên c <=5).

- Cập nhật biến đếm sau mỗi lần thực hiện câu lệnh lặp, thường là **tăng lên 1** (như ví dụ trên c++;).

Số câu lệnh lặp sẽ là = giá trị cuối - giá trị đầu + 1.

Ví dụ 3.5. In từ 1 đến N

Viết chương trình in các số từ 1 đến N.

Input: Dòng duy nhất chứa số nguyên N thỏa $1 \leq N \leq 1000$.

Output: In ra giá trị theo yêu cầu, mỗi số in trên một dòng.

Sample Output 1

5

Sample Input 2

1

2

3

4

5

Lời giải.

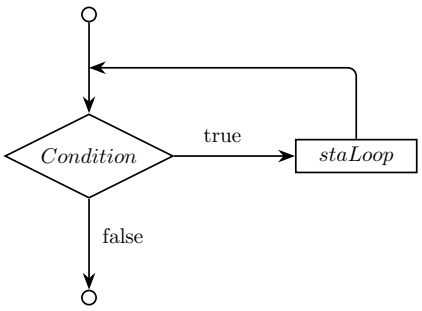
Câu lệnh lặp của bài toán này được tổng quát hóa là cout << i với $i = 1..n$, ở đây i đóng vai trò tham gia trong câu lệnh lặp và làm biến đếm vòng lặp

```
#include <bits/stdc++.h>
using namespace std;

int main() {
   int n; cin >> n;
   for (int i=1; i<=n; i++)
      cout << i << endl;
   return 0;
}
```

3.1.5 Lệnh lặp while

Cấu trúc lệnh lặp với số lần lặp không xác định, cho nên việc điều khiển lặp sẽ phụ thuộc vào biểu thức điều kiện. Sơ đồ khối của lệnh while với điều kiện trước như sau:



Hình 3.6: Sơ đồ khối lệnh while trong C++

Cú pháp:

```
while (Condition){
   staLoop;
}
```

Hoạt động: Câu lệnh lặp sẽ được thực hiện trong khi biểu thức điều kiện (Condition) có giá trị đúng (true). Điểm quan trọng nhất của cấu trúc lệnh này là bạn phải điều khiển được biểu thức điều kiện của mình để sao cho sau một số lần lặp nhất định thì phải kết thúc vòng lặp. Như vậy, trong khối lệnh lặp (staLoop) phải có ít nhất một phép toán tác động đến biểu thức điều kiện. Xét bài toán sau:

Ví dụ 3.6. Tổng các chữ số

Hãy lập trình tính tổng các chữ số của một số nguyên n cho trước.

Input: Dòng duy nhất chứa số nguyên n thỏa $|n| \leq 10^9$.

Output: In ra tổng cần tính.

Sample Output 1

34579

Sample Input 2

28

Lời giải.

Bài toán được giải bằng cách tách dần các chữ số của số nguyên n và cộng vào một tổng cho trước. Việc tách dần các chữ dễ dàng bằng việc tách hàng đơn vị dựa vào phép toán modulo cho 10. Sau mỗi lần tách ta giảm n xuống 10 đơn vị và lặp lại quy trình cho đến khi n bằng 0.

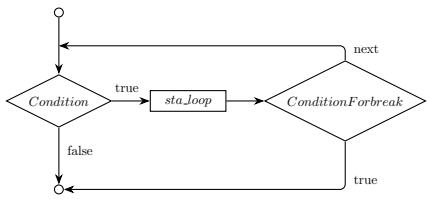
```
#include <bits/stdc++.h>
using namespace std;

int main() {
   int n; cin >> n;
   int s=0;
   while (n!=0){
      s = s + n%10;
      n = n/10;
   }
   cout << s;
   return 0;
}
```

3.1.6 Lệnh break, continue

Để sử dụng một cách tinh tế câu lệnh lặp với số lần lặp không xác định, biểu thức điều kiện là cực kỳ quan trọng. Bạn có thể sử dụng lệnh break hoặc continue một cách linh hoạt để hỗ trợ biểu thức điều kiện. Có thể phân chia như sau: Biểu thức điều kiện đầu của while là điều kiện mềm, còn điều kiện để dùng break gọi là điều kiện cứng (ngắt cứng).

Sơ đồ khối của lệnh break với điều kiện trước như sau:



Hình 3.7: Sơ đồ khối lệnh break while trong C++

Hoạt động: Vòng lặp được ngắt một cách linh hoạt bằng biểu thức điều kiện (Condititon) của vòng lặp

while và còn có thể ngắt vòng lặp bởi điều kiện của lệnh break. Xét ví dụ bài toán sau:

Ví dụ 3.7. Ốc sên trèo cây

Con ốc sên đang ở gốc của một cái cây độ cao h mét tính từ gốc. Ốc sên muốn bò lên ngọn cây để ăn lá. Ban ngày sên leo lên được độ cao a mét, nhưng ban đêm nó lại trụt xuống b mét. Hỏi nó cần bao nhiêu ngày để ăn được lá non.

Input: Dòng duy nhất chứa ba số nguyên h, a, b thỏa $1 \leq b \leq a \leq h \leq 10^9$.

Output: In ra số ngày cần tìm, nếu không có đáp số thì in -1.

Sample Input 1

5 2 1

Sample Output 1

4

Link nộp bài: https://coder.husc.edu.vn/problem/pb0051

Lời giải.

Một bài toán tưởng như đơn giản nhưng nó sẽ phức tạp khi $a = b$ khi đó ta rơi vào trường hợp lặp vô tận, vậy một biểu thức điều kiện cho ngắt cứng là break giúp ta giải quyết được trường hợp này.

```
#include <bits/stdc++.h>
using namespace std;

int main() {
   int h, a, b; cin >> h >> a  >> b;
   if ((a<=b) and (a<h)) cout <<"-1";
   else{
      int d =0, t=0;
      while (t < h){
         t += a;  d += 1;
         if (t >= h) break;
         t -= b;
      }
      cout<<d;
   }
   return 0;
}
```

3.2 Bài tập chương 3

3.2.1 Bài tập về lệnh rẽ nhánh

Bài toán 3.1. Tính giá trị A

Xác định giá trị của công thức $\mathrm{A} = \begin{cases} x^3 + 5x & \text{nếu } x \geq 10 \\ x^2 - 2x + 4 & \text{nếu } x < 10 \end{cases}$

Input: Dòng duy nhất là số nguyên x thỏa $|x| \leq 10^5$.

Output: In ra kết quả cần tính.

Sample Input 1

2

Sample Output 1

4

Sample Input 2

20

Sample Output 2

8100

Link nộp bài: https://coder.husc.edu.vn/problem/pb0011

Bài toán 3.2. Tính giá trị B

Xác định giá trị của công thức B $= \begin{cases} x^3 + 2x^2 + 5x & \text{nếu } x > 3 \\ x^2 - 2x + 4 & \text{nếu } x \in [1..3] \\ 5x - 8 & \text{nếu } x < 1 \end{cases}$

Input: Dòng duy nhất là số nguyên x thỏa $|x| \leq 10^5$.

Output: In ra kết quả cần tính.

Sample Input 1

2

Sample Output 1

4

Sample Input 2

20

Sample Output 2

8900

Link nộp bài: https://coder.husc.edu.vn/problem/pb0012

Bài toán 3.3. Tính giá trị C

Xác định giá trị của công thức C = $\begin{cases} x^2 + \sqrt{x} + 1 & \text{nếu } x > 0 \\ \dfrac{x^3 + 2x + 1}{x + 3} & \text{nếu ngược lại.} \end{cases}$

Input: Dòng duy nhất là số thực x.

Output: In ra kết quả cần tính với sáu chữ số sau dấu chấm thập phân. Trường hợp không tính được với biểu thức đã cho in thông báo **Div by zero**.

Sample Input 1

10.5

Sample Output 1

114.490370

Sample Input 2

-3.0

Sample Output 2

Div by zero

Link nộp bài: https://coder.husc.edu.vn/problem/pb0013

Bài toán 3.4. Công thức Heron

Viết chương trình tính diện tích của một tam giác với ba cạnh a, b, c cho trước.

Input: Dòng duy nhất chứa ba số thực a, b, c là ba cạnh của tam giác.

Output: In ra diện tích cần tính với bốn chữ số sau dấu chấm thập phân. Trường hợp không thỏa mãn điều kiện để tính diện tích tam giác in ra thông báo **No Solution**.

Sample Input 1

3 4 5

Sample Output 1

6.0000

Link nộp bài: https://coder.husc.edu.vn/problem/pb0015

Bài toán 3.5. Giải phương trình bậc hai

Viết chương trình giải phương trình bậc hai $ax^2 + bx + c = 0$ với điều kiện $a \neq 0$.

Input: Dòng duy nhất chứa ba số thực a, b, c.

Output: Dòng thứ nhất in nghiệm x_1 với bốn chữ số sau dấu chấm thập phân.

Dòng thứ hai in nghiệm x_2 nếu có với bốn chữ số sau dấu chấm thập phân.

Nếu vô nghiệm thì chỉ in một dòng thông báo **No Solution**.

Sample Input 1

9 9 -23

Sample Output 1

1.1750

-2.1750

Sample Input 2

1 2 1

Sample Output 2

-1.0000

Link nộp bài: https://coder.husc.edu.vn/problem/pb0016

Bài toán 3.6. Bội số của nhau

Cho hai số nguyên m, n, viết chương trình kiểm tra xem m là bội số của n hoặc ngược lại hay không?

Input: Dòng duy nhất chứa hai số nguyên m, n thỏa $1 \leq m, n \leq 10^{18}$.

Output: In ra Yes nếu thỏa và No nếu ngược lại.

Sample Input 1

6 24

Sample Output 1

Yes

Sample Input 2

235453543543 7834546466

Sample Output 2

No

Link nộp bài: https://coder.husc.edu.vn/problem/pb0017

Bài toán 3.7. Phân biệt tam giác bản dễ

Cho tam giác với ba cạnh a, b, c và chỉ có ba dạng là: Tam giác cân, tam giác đều và tam giác thường. Hãy lập trình phân biệt ba loại tam giác trên.

Input: Dòng duy nhất chứa số ba số nguyên a, b, c là ba cạnh của tam giác thỏa $1 \leq a, b, c \leq 10^9$.

Output: In ra Tam giac can hoặc Tam giac deu hoặc Tam giac thuong tương ứng với ba cạnh. Dữ liệu đảm bảo có nghiệm.

Sample Input 1

5 6 7

Sample Output 1

Tam giac thuong

Sample Input 2

1 1 1

Sample Output 2

Tam giac deu

Link nộp bài: https://coder.husc.edu.vn/problem/pb0019

Bài toán 3.8. Phân biệt tam giác bản khó

Cho tam giác với ba cạnh a, b, c và chỉ có ba dạng là: Tam giác cân, tam giác đều và tam giác thường. Hãy lập trình phân biệt ba loại tam giác trên.

Input: Dòng duy nhất chứa ba số thực a, b, c là ba cạnh của tam giác với độ chính xác 9 chữ số sau dấu chấm thập phân.

Output: In ra Tam giac can hoặc Tam giac deu hoặc Tam giac thuong tương ứng với ba cạnh. Dữ liệu đảm bảo có nghiệm.

Sample Input 1

3.1234567899 3.1234567899 4.02

Sample Output 1

Tam giac can

Sample Input 2

1 1 1

Sample Output 2

Tam giac deu

Link nộp bài: https://coder.husc.edu.vn/problem/pb0020

Bài toán 3.9. Đem tủ vào nhà

Ba Bi mới mua một cái tủ đựng sách vở với kích thước ba chiều là $a \times b \times c$. Ba hỏi Bi liệu có đem cái tủ vào nhà được hay không khi cửa nhà mình có kích thước hai chiều là $x \times y$. Khi đưa vào cho phép xoay tủ theo các chiều khác nhau.

Input: Dòng duy nhất chứa năm số nguyên dương a, b, c, x, y thỏa $1 \leq a, b, c, x, y \leq 300$.

Output: In ra Yes nếu thỏa và No nếu ngược lại.

Sample Input 1

4 5 6 10 20

Sample Output 1

Yes

Sample Input 2

120 160 50 80 100

Sample Output 2

No

Link nộp bài: https://coder.husc.edu.vn/problem/pb0021

Bài toán 3.10. Ít nhất một điều kiện

Với số nguyên n cho trước, hãy viết chương trình in ra YES nếu thỏa ít nhất một trong các điều kiện sau và NO nếu ngược lại.

- Nếu n là số lẻ.

- Nếu n lớn hơn 0 và có ba chữ số.

Input: Dòng duy nhất chứa số nguyên n thỏa $|n| \leq 10^9$.

Output: In ra YES nếu thỏa và NO nếu ngược lại.

Sample Input 1

-6

Sample Output 1

NO

Sample Input 2

113

Sample Output 2

YES

Link nộp bài: https://coder.husc.edu.vn/problem/pb0023

Bài toán 3.11. Kiểm tra thuộc khoảng, đoạn

Cho số thực x, hãy xác định xem x thuộc đoạn, khoảng nào trong:$[0, 25], (25, 50], (50, 75], (75, 100]$.

Chú ý: Ký hiệu '(' đại diện cho lớn hơn, ký hiệu ')' đại diện cho nhỏ hơn, ký hiệu '[' đại diện cho lớn hơn hoặc bằng, ký hiệu ']' đại diện cho nhỏ hơn hoặc bằng.

Input: Dòng duy nhất chứa số thực x thỏa $-1000 \leq x \leq 1000$.

Output: In ra theo yêu cầu theo mẫu như ví dụ.

Sample Input 1

25.1

Sample Output 1

Interval (25,50]

Sample Input 2

25.0

Sample Output 2

Interval [0,25]

Sample Input 3

100.5

Sample Output 3

Out of Intervals

Link nộp bài: https://coder.husc.edu.vn/problem/pb0024

Bài toán 3.12. Hai đoạn chồng nhau

Cho hai đoạn $[L_1, R_1], [L_2, R_2]$. Viết chương trình tìm đoạn giao của chúng.

Input: Dòng duy nhất chứa bốn số nguyên L_1, R_1, L_2, R_2 thỏa $1 \leq L_1 \leq R_1, L_2 \leq R_2 \leq 10^9$.

Output: Nếu có giao nhau giữa hai đoạn này thì in ranh giới của nó, ngược lại thì in -1.

Sample Input 1

1 15 5 27

Sample Output 1

5 15

Sample Input 2

2 5 6 12

Sample Output 2

-1

Link nộp bài: https://coder.husc.edu.vn/problem/pb0025

Bài toán 3.13. Đổi số ra giờ

Viết chương trình nhập vào một số n là số giây có giá trị từ 0 đến 86399. Hãy lập trình đổi giây đó thành dạng "gio:phut:giay", ví dụ: 77 thành 00 : 01 : 17

Input: Dòng duy nhất chứa số nguyên n như yêu cầu.

Output: In ra định dạng theo yêu cầu chuyển đổi.

Sample Input 1

77

Sample Output 1

00:01:17

Sample Input 2

43

Sample Output 2

00:00:43

Link nộp bài: https://coder.husc.edu.vn/problem/pb0026

Bài toán 3.14. Hai chữ số cuối

Viết chương trình xác định hai chữ số cuối của phép tính $X = a \times b \times c \times d$ với bốn số nguyên a, b, c, d cho trước.

Input: Dòng duy nhất chứa bốn số nguyên a, b, c, d thỏa $2 \leq a, b, c, d \leq 10^9$.

Output: In kết quả cần tính.

Sample Input 1

3 9 9 9

Sample Output 1

87

Sample Input 2

10 10 10 10

Sample Output 2

00

Link nộp bài: https://coder.husc.edu.vn/problem/pb0027

Bài toán 3.15. Thỏ trong hang

Nguyên lý Dirichlet được phát biểu "nếu xếp nhiều hơn $n+1$ đối tượng vào n cái hộp thì tồn tại ít nhất 1 hộp chứa không ít hơn 2 đối tượng". Áp dụng nguyên lý đó để giải bài toán sau:

Giả sử có n cái hang và m con thỏ. Tính số thỏ tối đa được bảo đảm ở cùng một hang.

Input: Dòng duy nhất chứa hai số nguyên dương n, m thỏa $1 \leq n \leq m \leq 10^9$.

Output: In ra số lượng thỏ tối đa được đảm bảo nằm trong cùng một hang.

Sample Input 1

2 3

Sample Output 1

2

Sample Input 2

98 234567

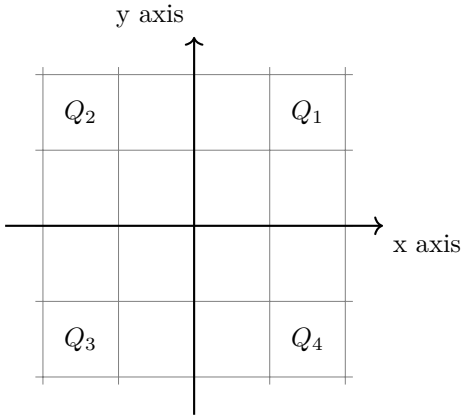
Sample Output 2

2394

Link nộp bài: https://coder.husc.edu.vn/problem/pb0028

Bài toán 3.16. Vị trí của điểm

Cho hai số thực x, y là tọa độ của một điểm trên mặt phẳng Oxy. Hãy xác định xem điểm đó nằm ở vị trí nào trên mặt phẳng, Các vị trí đánh nhãn như hình vẽ sau:



Input: Dòng duy nhất chứa hai số thực x, y thỏa $-1000 \leq x, y \leq 1000$, các số thực với số chữ số thập phân nhỏ hơn bằng 4 chữ số.

Output: In ra vị trí $Q1, Q2, Q3, Q4$ của điểm như hình vẽ. Nếu chúng không thuộc các phần trên mà nằm trên gốc thì in Origem, hoặc trên đường thẳng tọa độ thì in Eixo X, Eixo Y tương ứng.

Sample Input 1

4.5 -2.2

Sample Output 1

Q4

Sample Input 2

0 0

Sample Output 2

Origem

Link nộp bài: https://coder.husc.edu.vn/problem/pb0029

Bài toán 3.17. Thuê phòng khách sạn

Lớp Bi tổ chức đi thăm quan du lịch vào mùa hè 2 ngày và phải trọ lại ở khách sạn. Lớp có n bạn trai và m bạn gái, mỗi phòng của khách sạn có k giường, mỗi bạn được tiêu chuẩn 1 giường. Cô Bi không biết phải thuê bao nhiêu phòng để không có bạn trai với bạn gái nào ở cùng phòng.

Hãy lập trình giúp cô Bi.

Input: Dòng duy nhất là ba số nguyên n, m, k thỏa $0 \leq n, m \leq 100; 1 \leq k \leq 100$. Dữ liệu cho đảm bảo có kết quả.

Output: In kết quả cần tìm.

Sample Input 1

6 12 3

Sample Output 1

6

Link nộp bài: https://coder.husc.edu.vn/problem/pb0030

Bài toán 3.18. Cân bằng chân bàn

Bi tự đóng một cái bàn bốn chân, tuy nhiên sau khi đóng và sử dụng thì Bi cảm thấy cái bàn của mình không cân bằng nên cô ấy lấy thước đo lại bốn cái chân của bàn và thu được bốn số nguyên a_1, a_2, a_3, a_4. Bi quyết định nối thêm một trong các chân bàn bằng một đoạn gỗ có chiều dài b.

Câu hỏi đặt ra liệu Bi có thể làm cho cái bàn của mình bằng phẳng hay không?

Input: Dòng đầu tiên chứa số T là số testcase thỏa $1 \leq T \leq 10$.

T dòng tiếp theo, mỗi dòng chứa năm số nguyên a_1, a_2, a_3, a_4, b thỏa $1 \leq a_i, b \leq 100$.

Output: Ứng với mỗi testcase in ra Yes nếu thỏa và No nếu ngược lại, mỗi testcase in trên một dòng.

Sample Input 1

3

10 10 10 10 5

13 13 5 13 8

50 42 42 50 8

Sample Output 1

Yes

Yes

No

Link nộp bài: https://coder.husc.edu.vn/problem/pb0031

3.2.2 Bài tập về lệnh lặp for

Bài toán 3.19. Đếm và liệt kê ước số

Hãy lập trình đếm và liệt kê các ước số của số nguyên dương n cho trước.

Input: Dòng duy nhất chứa số nguyên n thỏa $1 \leq n \leq 10^7$.

Output: Dòng thứ nhất in số lượng ước số của n.

Dòng thứ hai là các ước số của n mỗi số cách nhau ký tự trắng.

Sample Input 1

200

Sample Output 1

12

1 2 4 5 8 10 20 25 40 50 100 200

Link nộp bài: https://coder.husc.edu.vn/problem/pb0032

Bài toán 3.20. Liệt kê số chia hết

Hãy lập trình liệt kê các số từ 1 đến n thỏa mãn điều kiện chia hết cho p hoặc q.

Input: Dòng duy nhất là ba số nguyên n, p, q thỏa $1 \leq n \leq 10^7; 1 \leq p, q \leq n$.

Output: Liệt kê các số trên một dòng, các phần tử cách nhau dấu ký tự trắng.

Sample Input 1

10 3 7

Sample Output 1

3 6 7 9

Link nộp bài: https://coder.husc.edu.vn/problem/pb0033

Bài toán 3.21. Số đối xứng

Số nguyên dương n được gọi là số đối xứng nếu viết theo chiều ngược lại ta cũng được chính nó, ví dụ: $121, 11, 2332$. Hãy lập trình in ra Yes nếu n là số đối xứng và No nếu ngược lại.

Input: Dòng duy nhất chứa số nguyên n thỏa $1 \leq n \leq 10^{18}$.

Output: In ra theo yêu cầu.

Sample Input 1

1234

Sample Output 1

No

Sample Input 2

123454321

Sample Output 2

Yes

Link nộp bài: https://coder.husc.edu.vn/problem/pb0034

Bài toán 3.22. Vẽ chữ X

Vẽ chữ X như ví dụ minh họa với kích thước $n = 2$.

Input: Dòng duy nhất chứa số nguyên n thỏa $1 \leq n \leq 12$.

Output: In ra theo yêu cầu, tham khảo ví dụ mẫu.

Sample Input 1

2

Sample Output 1

```
* . . . *
. * . * .
. . * . .
. * . * .
* . . . *
```

Link nộp bài: https://coder.husc.edu.vn/problem/pb0035

Bài toán 3.23. Số biểu đồ

Dữ liệu đầu vào của bài toán được mô tả như sau:

- Dòng đầu tiên chứa ký hiệu s (gồm $+, -, , /$).

- Dòng thứ hai chứa số n.

- Dòng thứ ba chứa n số nguyên x_i.

Hãy lập trình ứng với mỗi số x_i trong n số trên in ra một dòng chứa ký hiệu s lặp lại x_i lần.

Input: Dữ liệu vào như mô tả của bài toán với ràng buộc $1 \leq n \leq 50; 1 \leq x_i \leq 100$.

Output: In ra theo yêu cầu.

Sample Input 1

+

3

1 2 3

Sample Output 1

+

++

+++

Link nộp bài: https://coder.husc.edu.vn/problem/pb0036

Bài toán 3.24. Tìm cặp số

Cho số nguyên n, viết chương trình tìm các cặp số nguyên a, b trong đoạn [1..200] thỏa công thức:

$$\frac{1}{a} + \frac{1}{b} = \frac{1}{n}$$

Input: Dòng duy nhất chứa số nguyên n thỏa $1 \leq n \leq 10$.

Output: In ra từng cặp a, b thỏa điều kiện, mỗi cặp in trên một dòng.

Sample Input 1

3

Sample Output 1

4 12

6 6

12 4

Link nộp bài: https://coder.husc.edu.vn/problem/pb0037

Bài toán 3.25. Kiểm tra số nguyên tố

Hãy lập trình kiểm tra xem với số nguyên dương n cho trước có phải là số nguyên tố hay không?

Input: Dòng duy nhất là số nguyên dương n thỏa $1 \leq n \leq 10^9$.

Output: In ra Yes nếu đúng và No nếu ngược lại.

Sample Input 1

50

Sample Output 1

No

Sample Input 2

11

Sample Output 2

Yes

Link nộp bài: https://coder.husc.edu.vn/problem/pb0038

Bài toán 3.26. Gà và chó

Có một bài toán dân gian khá thú vị như sau:

Vừa gà vừa chó

Bó lại cho tròn

Ba mươi sáu con

Một trăm chân chẵn

Hỏi mấy gà, mấy chó?

Bài toán lập trình dành cho các bạn như sau: Cho hai số nguyên m, n, với m là tổng số gà với chó, n là tổng số chân. Viết chương trình in ra số gà và chó tương ứng.

Input: Dòng duy nhất chứa hai số nguyên m, n thỏa $1 \leq m < 3.10^4; m \leq n \leq 2.10^5$.

Output: In ra số gà và chó cần tìm theo mẫu như ví dụ. Nếu không có nghiệm thì in -1.

Sample Input 1

36 100

Sample Output 1

Ga = 22

Cho = 14

Link nộp bài: https://coder.husc.edu.vn/problem/pb0039

Bài toán 3.27. Tổng từ 1 đến n

Viết chương trình tính tổng $S = 1 + 2 + \ldots + n$, với n nguyên dương cho trước.

Input: Dòng duy nhất chứa số nguyên n thỏa $1 \leq n \leq 10^7$.

Output: In ra tổng cần tính.

Sample Input 1

10

Sample Output 1

55

Link nộp bài: https://coder.husc.edu.vn/problem/pb0040

Bài toán 3.28. Giai thừa kép

Số giai thừa kép được định nghĩa như sau:

$$\text{n!!} = \begin{cases} 1 & \text{nếu n=0 hoặc n=1} \\ 1 \times 3 \times 5 \times \ldots \times n & \text{nếu n lẻ} \\ 2 \times 4 \times 6 \times \ldots \times n & \text{nếu n chẵn} \end{cases}$$

Hãy viết chương trình tính số giai thừa kép trên.

Input: Dòng duy nhất chứa số nguyên n thỏa $-1 \leq n \leq 30$.

Output: In ra kết quả cần tính.

Sample Input 1

7

Sample Output 1

105

Sample Input 2

4

Sample Output 2

8

Link nộp bài: https://coder.husc.edu.vn/problem/pb0041

Bài toán 3.29. Số Harmonic

Số Harmonic được tính bằng công thức:

$$H_n = \sum_{i=1}^{n} \frac{1}{i}$$

Với n nguyên dương cho trước, hãy lập trình tính giá trị của số Harmonic

Input: Dòng duy nhất chứa số nguyên n thỏa $1 \leq n \leq 10^6$.

Output: In ra tổng cần tính với 6 chữ số sau dấu chấm thập phân.

Sample Input 1

10

Sample Output 1

2.928968

Link nộp bài: https://coder.husc.edu.vn/problem/pb0042

Bài toán 3.30. Số có tổng các ước số lớn hơn chính nó

Bi rất thích số học và nhận thấy số 12 có các ước số $1, 2, 3, 4$ và 6. Thực hiện phép cộng các ước số trên được tổng là 16 và là số lớn hơn 12. Tuy nhiên lại có những số không thỏa tính chất trên, ví dụ số 21 có tổng các ước là 11. Cho số nguyên N, hãy lập trình giúp Bi kiểm tra số N nào thỏa điều kiện trên.

Input: Dòng duy nhất chứa số nguyên N thỏa $1 \leq N \leq 10^9$.

Output: In ra YES nếu thỏa và NO trong trường hợp ngược lại.

Sample Input 1

12

Sample Output 1

YES

Sample Input 2

21

Sample Output 2

NO

Link nộp bài: https://coder.husc.edu.vn/problem/pb0043

3.2.3 Bài tập về lệnh lặp while

Bài toán 3.31. Đổi số hệ 10 sang hệ 16

Hãy lập trình đổi một số nguyên dương n từ hệ cơ số 10 sang số ở hệ cơ số 16 (Hệ cơ số 16 gồm các ký tự: $0 \dots 9, A, B, C, D, E, F$)

Input: Dòng duy nhất chứa số nguyên n thỏa $1 \leq n \leq 10^9$.

Output: In ra số cần đổi.

Sample Input 1

47

Sample Output 1

2F

Sample Input 2

10

Sample Output 2

A

Link nộp bài: https://coder.husc.edu.vn/problem/pb0044

Bài toán 3.32. Tỷ lệ tổng và tích các chữ số

Lập trình tìm tỷ số giữa tích và tổng của các chữ số của một số tự nhiên cho trước. Ví dụ $n = 36$ ta có tích các chữ số là 18, tổng các chữ số là 9 nên tỷ số là 2.000.

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $n \leq 10^{18}$.

Output: In ra tỷ số cần tính với 3 chữ số sau dấu chấm thập phân.

Sample Input 1

36

Sample Output 1

2.000

Sample Input 2

199999999

Sample Output 2

589681.110

Link nộp bài: https://coder.husc.edu.vn/problem/pb0045

Bài toán 3.33. Phỏng đoán Collatz

Phỏng đoán Collatz còn được gọi là phỏng đoán $3n+1$ là một phỏng đoán rất nổi tiếng và lâu đời trong toán học thể hiện như sau: Lấy số tự nhiên n bất kỳ, nếu n chẵn, chia cho 2 để được $\frac{n}{2}$ và nếu n là số lẻ lớn hơn 1, nhân nó với 3 và thêm 1 để được $3n+1$. Lặp lại quá trình này để nhận được một dãy số tự nhiên được gọi là dãy Hailstone, ví dụ $n=3$ ta có dãy $a=\{3,10,5,16,8,4,2,1\}$.

Phỏng đoán cho rằng bạn bắt đầu bằng bất kỳ số nào bạn luôn đạt 1. Hãy viết một chương trình tính toán độ dài của dãy Hailstone với số nguyên dương n cho trước.

Input: Dòng duy nhất chứa số nguyên n thỏa $1 \leq n \leq 100$.

Output: In ra độ dài dãy cần tính.

Sample Input 1

8

Sample Output 1

4

Sample Input 2

3

Sample Output 2

8

Link nộp bài: https://coder.husc.edu.vn/problem/pb0046

Bài toán 3.34. Tính tổng lương công nhân

Hãy lập trình tính tổng lương cần trả cho các công nhân của công ty bằng cách mô tả phần input, output sau:

Input: Gồm nhiều dòng, mỗi dòng là lương cần trả cho mỗi người. Dòng có giá trị -1 dùng để kết thúc việc nhập dữ liệu.

Output: In ra tổng cần tính.

Sample Input 1

12345

54321

67890

98765

-1

Sample Output 1

233321

Link nộp bài: https://coder.husc.edu.vn/problem/pb0047

Bài toán 3.35. Phân tích ra thừa số nguyên tố

Lập trình nhập vào một số tự nhiên n và phân tích số đó thành các thừa số nguyên tố, ví dụ $n = 100 = 2.2.5.5$

Input: Dòng duy nhất chứa số nguyên n thỏa $1 \leq n \leq 10^9$.

Output: In ra các thừa số nguyên tố của, mỗi số cách nhau ký tự trắng.

Sample Input 1

100

Sample Output 1

2 2 5 5

Sample Input 2

6

Sample Output 2

2 3

Link nộp bài: https://coder.husc.edu.vn/problem/pb0048

Bài toán 3.36. Tổng chữ số chia hết bởi n

Gọi $S(n)$ là tổng của các chữ số của số nguyên n, ví dụ $S(12) = 1 + 2 = 3$. Hãy lập trình giúp Bi kiểm tra xem $S(n)$ có được chia hết bởi n hay không?

Input: Dòng duy nhất chứa số nguyên n thỏa $1 \leq n \leq 10^{19}$.

Output: In ra Yes nếu thỏa và No nếu ngược lại.

Sample Input 1

12

Sample Output 1

Yes

Sample Input 2

101

Sample Output 2

No

Link nộp bài: https://coder.husc.edu.vn/problem/pb0049

Bài toán 3.37. Tính tổng gần đúng

Lập trình tính tổng $S = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!}$, với độ chính xác $\frac{x^n}{n!} < 10^{-9}$, x là số thực nhập từ bàn phím.

Input: Dòng duy nhất chứa số thực x thỏa $|x| \leq 30$.

Output: In ra số cần tính với 4 chữ số sau dấu chấm thập phân.

Sample Input 1

6.07

Sample Output 1

432.6807

Link nộp bài: https://coder.husc.edu.vn/problem/pb0050

Bài toán 3.38. Lãi suất kép

Bánh rất quý trọng đồng tiền và đã gửi n đô la vào tài khoản của mình tại một ngân hàng với lãi suất kép $p\%$ một tháng. Hỏi để có được số tiền ít nhất là m thì Bánh cần gửi bao nhiêu tháng?

Input: Dòng duy nhất chứa ba số thực n, m, p thỏa $n \leq m$.

Output: In ra số tháng cần gửi.

Sample Input 1

1000 1.1 1200

Sample Output 1

17

Link nộp bài: https://coder.husc.edu.vn/problem/pb0052

Chương 4

Mảng và vector

4.1 Tóm tắt lý thuyết

Kiểu dữ liệu Mảng (array) được xem như là một trong những kiểu dữ liệu quan trọng và ứng dụng rất nhiều trong các bài toán lập trình. Nó cho phép lưu trữ được một tập N các đối tượng có cùng kiểu dữ liệu trong bộ nhớ. Để từ đó ta có thể tính toán, xử lý dữ liệu của N đối tượng trên.

Hầu hết các ngôn ngữ lập trình đều có cơ chế cho phép người lập trình khai báo kiểu mảng dựa vào các kiểu dữ liệu cơ bản như số nguyên, số thực, ký tự hoặc kiểu có cấu trúc (struct, pair, tuple).

Kiểu mảng được khai báo ấn định với số phần tử xác định nên dễ dàng xác định được kích thước bộ nhớ và đây cũng chính là nhược điểm của mảng. Để khắc phục nhược điểm này người ta đưa ra cơ chế khai thác bộ nhớ động với biến con trỏ và các kiểu dữ liệu như danh sách liên kết.

Ưu điểm nổi trội của kiểu mảng là tốc độ truy cập các phần tử với độ phức tạp $\mathcal{O}(1)$ phù hợp với việc lưu trữ dữ liệu hỗ trợ cho phép tính toán. Các dạng thuật toán cơ bản như sắp xếp và tìm kiếm dữ liệu và thống kê được xem là cơ bản trên kiểu dữ liệu mảng.

C++ còn cung cấp cho người lập trình lớp vectơ là lớp chứa (container) lưu trữ dữ liệu kiểu danh sách tuyến tính theo mô hình cấp phát động nên linh hoạt hơn mảng tĩnh. Lập trình với vector rất hiệu quả vì là lớp mẫu nên có rất nhiều phương thức dựng sẵn làm việc được với nhiều kiểu dữ liệu khác nhau. Có thể tạo các mảng động mà không cần phải cấp phát và thu hồi vùng nhớ bằng cách sử dụng toán tử new và delete khi dùng với con trỏ. Vector là một lớp chứa cung cấp khả năng sử dụng mảng mềm dẻo, có kiểm soát phạm vi truy cập tốt với kích thước tùy ý.

4.1.1 Định nghĩa và khai báo mảng một chiều

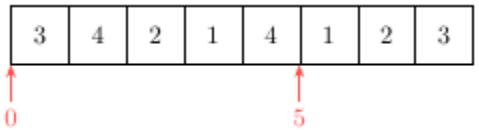
Định nghĩa:

Mảng là kiểu dữ liệu cho phép lưu trữ một tập N phần tử cùng kiểu dữ liệu liên tiếp nhau trong bộ nhớ, do các phần tử có cùng kiểu nên dễ dàng tính được kích thước bộ nhớ cần cung cấp cho mảng. Các phần tử của mảng được tham chiếu (truy cập) bằng chỉ mục, địa chỉ đầu tiên là 0.

Ví dụ 4.1. Bộ nhớ của mảng

Đoạn mã sau thể hiện trong bộ nhớ như hình vẽ.

```
#include <iostream>
using namespace std;
int main()
{
   int A[] ={3, 4, 2, 1, 4, 1, 2, 3};
   cout <<A[0]<<" "<<A[5]<<endl;
   return 0;
}
```



Hình 4.1: Mảng và chỉ số

Chương trình trên in ra hai giá trị $3, 1$ của mảng tại hai vị trí index là 0 và 5.

Khai báo và khởi tạo

Cú pháp: <TYPE> arrayName[nmax];

Trong đó:

- TYPE: Là các kiểu dữ liệu cơ bản, ví dụ như : int, long long, double.
- arrayName: Tên biến mảng.
- nmax: Số phần tử tối đa của mảng cần khai báo.

Khai báo tĩnh, toàn cục

Là khai báo ngoài hàm main(), ưu điểm là được khởi tạo giá trị ban đầu.

Ví dụ 4.2. Khai báo và khởi tạo giá trị đầu

```
#include <iostream>
using namespace std;
const int nmax=100+1;
int dp[nmax];
bool vis[nmax];
struct point { int x,y; };
point ps[nmax];

int main()
{
   // Các phần tử của mảng dp được khởi tạo với giá trị mặc định là 0.
```

```
12    for (int i=0; i<nmax; i++)
13       cout<< dp[i] <<" ";
14    cout<<endl;
15    // Các phần tử mảng vis có giá trị false.
16    for (int i=0; i<nmax; i++)
17       cout<< vis[i] <<" ";
18    cout<<endl;
19    // Các tọa độ điểm ps có giá trị trường x == 0 và y == 0.
20    for (int i=0; i<nmax; i++)
21       cout<< ps[i].x <<" " <<ps[i].y<<endl;
22    return 0;
23 }
```

Chú ý: Do biết trước số lượng các phần tử nên ta khai báo hằng cho kích thước mảng: const int nmax=100+1;

Ở đây có cộng thêm 1 tránh trường hợp truy cập ngoài vùng chỉ mục theo thói quen một số người đánh chỉ mục từ 1 khi lập trình.

Khai báo tĩnh, địa phương

Là khai báo trong hàm và không được khởi tạo giá trị đầu. Do vậy, tránh trường hợp rác người ta vừa khai báo vừa gán giá trị.

Cú pháp: <TYPE> arrayName[nmax] = {};

Ví dụ 4.3. Khai báo và không khởi tạo giá trị đầu

```
1  #include <iostream>
2  using namespace std;
3  const int nmax=100+1;
4  int main()
5  {
6     int A[nmax] ; // các phần tử mảng A có giá trị rác (garbage)
7     for (int i=0; i<nmax; i++)
8        cout<< A[i] <<" ";
9     cout<<endl;
10
11    int B[nmax] = { } ; // các phần tử mảng B có giá trị là 0.
12    for (int i=0; i<nmax; i++)
13       cout<< B[i] <<" ";
14
15    return 0;
16 }
```

Khai báo mảng động, địa phương

Là cách khai thác vùng nhớ dynamic (heap) của bộ nhớ khác với vùng nhớ static hạn chế ở trên. Để sử dụng được vùng nhớ này C, C++ cung cấp cơ chế rất đặc biệt đó là biến con trỏ. Tuy nhiên để đơn giản hóa tránh sự phức tạp khi sử dụng biến con trỏ C++ đưa ra cú pháp sau cho việc tạo ra một mảng động với kích thước được cấp phát khi chạy chương trình (khai báo trong hàm main()).

Ví dụ 4.4. Khai báo và khởi tạo giá trị đầu mảng động

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

int main()
{
   int n; cin >> n; //n là kích thước của mảng.
   int A[n] ; //cấp phát mảng n phần tử số nguyên.
   //gán giá trị cho mảng bằng hàm fill().
   fill(A, A + n, 0);
   for (int i=0; i<nmax; i++)
      cout<< A[i] <<" ";

   return 0;
}
```

Mảng động phổ biến nhất trong C ++ là cấu trúc vectơ, có thể được sử dụng gần giống như một mảng thông thường.

Khai báo vector

vector<TYPE> vecname;

hoặc

vector<TYPE> vecname(n);

hoặc

vector<TYPE> vecname(n, init_value);

Ví dụ 4.5. Khai báo vector các số nguyên

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

int main()
{

   int n; cin >> n; //n là kích thước của mảng.
   vector<int> B(n);
```

```
9     vector<int> A;
10    //Gán giá trị cho vector A, B
11    for (int i = 0; i < n; i++){
12       A.push_back(i);
13       B[i]=i*i;
14    }
15    for (int i=0; i<n; i++)
16       cout<< A[i] <<" ";
17    cout<<endl;
18    for (int i=0; i<n; i++)
19       cout<< B[i] <<" ";
20    cout<<endl;
21    return 0;
22 }
```

4.1.2 Truy cập, xuất/nhập và duyệt mảng một chiều

Truy cập

Mảng là cấu trúc tuyến tính, các phần tử đều được đánh chỉ mục cho nên dễ dàng truy cập các phần tử với thời gian $\mathcal{O}(1)$.

Cú pháp: arrayName[idx]; trong đó idx là chỉ mục nhận giá trị từ 0 đến $n-1$.

Ví dụ 4.6. Liệt kê các số chia hết cho 2 trong dãy số $A = \{1, 3, 4, 2, 5, 7, 6, 6, 9\}$.

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6     int A[]={1, 3, 4, 2, 5, 7, 6, 6, 9} ; // Vừa khai báo vừa gán giá trị.
7     int size = sizeof(A)/sizeof(A[0]); //Tính số phần tử của mảng.
8
9     for (int i=0; i<size; i++)
10       if (A[i] %2 ==0)
11          cout<< A[i] <<" ";
12    cout<<endl;
13    return 0;
14 }
```

Chú ý: Có thể dễ dàng tham số hóa tên mảng trong một macro để tính kích thước mảng bất kỳ.

```
1  #define NELEMS(x)  (sizeof(x) / sizeof((x)[0]))
2  int a[17];
3  size_t n = NELEMS(a);
```

Nhập, xuất mảng và duyệt mảng

Để dễ dàng tiếp cận với hai thao tác này, ta thực hành giải bài toán ví dụ sau:

Ví dụ 4.7. Liệt kê số

Viết chương trình liệt kê các phần tử trong dãy $A = \{a_1, a_2, \ldots, a_n\}$ có giá trị lớn hơn bằng 5 và nhỏ hơn bằng 7.

Input: Dòng thứ nhất chứa số nguyên dương n là kích thước dãy thỏa $1 \leq n \leq 10^5$. Dòng tiếp theo chứa các số nguyên a_i của dãy. Dữ kiện đảm bảo có kết quả.

Output: In ra số phần từ thỏa điều kiện.

Sample Input 1

7

1 2 3 4 5 6 7

Sample Output 1

5 6 7

Lời giải 1: Sử dụng mảng tĩnh, toàn cục

```
#include <iostream>
using namespace std;
const int nmax=100001;
int a[nmax];

int main()
{
   int n;
   cin >> n ;
   for (int i = 0 ; i < n ; i++)
      cin >> a[i] ;
   for (int i = 0 ; i < n ; i++ )
      if ((a[i] >= 5) && (a[i] <= 7 ))
         cout << a[i] << " " ;
   return 0;
}
```

Lời giải 2: Sử dụng mảng động

```
#include <iostream>
using namespace std;
int main()
{
   int n;
```

```
6     cin >> n ;
7     int a[n];
8     for (int i = 0 ; i < n ; i++)
9       cin >> a[i] ;
10    for (int i = 0 ; i < n ; i++ )
11      if ((a[i] >= 5) && (a[i] <= 7 ))
12        cout << a[i] << " " ;
13    return 0;
14  }
```

Truy cập vector

- Dùng phép toán tải bội [idx]: Trả về phần tử tại vị trí chỉ mục *idx*, ví dụ v[0] trả về phần tử đầu tiên của vector.
- Hàm at(idx): Trả về phần tử tại vị trí chỉ mục *idx* tương tự phép toán trên.
- Hàm front(): Trả về phần tử đầu vector.
- Hàm back(): Trả về phần tử cuối vector.

Xét ví dụ với mã nguồn sau:

```
1   #include <bits/stdc++.h>
2   using namespace std;
3   typedef vector<int> vi;
4   int main()
5   {
6     vi a;
7     //Điền dữ lệu vào vector theo thứ tự 10, 20, .., 100.
8     for (int i = 1; i <= 10; i++)    a.push_back(i * 10);
9     cout << "\nReference operator a[] : a[1] = " << a[1];
10    cout << "\nat : a.at(5) = " << a.at(5);
11    cout << "\nfront() : a.front() = " << a.front();
12    cout << "\nback() : a.back() = " << a.back();
13    return 0;
14  }
```

Output:

Reference operator a[] : a[1] = 20

at : a.at(5) = 60

front() : a.front() = 10

back() : a.back() = 100

Chú ý: Mặc dù làm việc với vector nhưng thói quen sử dụng phép toán [] như ở mảng vẫn được dùng nhiều ở vector.

Xuất/nhập và duyệt vector

- Hàm begin(): Trả về con trỏ lặp (iterator) trỏ đến phần tử đầu vector.
- Hàm end(): Trả về con trỏ lặp (iterator) trỏ đến phần tử cuối vector.
- Hàm rbegin(): Trả về con trỏ lặp thứ tự ngược (iterator) trỏ đến phần tử cuối vector.
- Hàm rend(): Trả về con trỏ lặp thứ tự ngược (iterator) trỏ đến phần tử đầu vector.

Xét ví dụ như mã nguồn sau:

```
#include <bits/stdc++.h>
using namespace std;
typedef vector<int> VI;
int   main()
{
   VI v;                        // Khai báo vector.
   VI::iterator  it;             // Khai báo iterator.
   VI::reverse_iterator  rit;    // Khai báo iterator ngược.
   for (int i = 1; i <= 5; i++) v.push_back(i);

   cout << "Output of begin and end: ";
   for (auto i = v.begin(); i != v.end(); ++i) {
      cout << *i << " ";
   }

   cout << "\nOutput of begin and end by iterator: ";
   for (it=v.begin();it!=v.end();++it){
      cout <<  *it  <<  " " ;
   }

   cout << "\nOutput of rbegin and rend: ";
   for (auto ir = v.rbegin(); ir != v.rend(); ++ir){
      cout << *ir << " ";
   }

   cout << "\nOutput of begin and end by reverse iterator: ";
   for (rit=v.rbegin();rit!=v.rend();rit++){
      cout <<  *rit  <<  " ";
   }

   return 0;
}
```

Output:

Output of begin and end: 1 2 3 4 5

Output of begin and end by iterator: 1 2 3 4 5

Output of rbegin and rend: 5 4 3 2 1

Output of begin and end by reverse iterator: 5 4 3 2 1

Chú ý: Không nên viết for (i=0;i<=v.size()-1;i++) Vì nếu vector v rỗng, v.size() là kiểu unsigned int, nên v.size()-1 sẽ bằng $2^{32} - 1$

Đối với bộ biên dịch C++ std11 trở lên, ta có thể duyệt vector bằng cú pháp sau: for (auto x : v) traiter(x);.

Ví dụ 4.8. Tính tổng

Viết chương trình nhập vào một dãy số nguyên $A = \{a_1, a_2, \ldots, a_n\}$, tính tổng và in kết quả.

```
#include <bits/stdc++.h>
using namespace std;
int main(){
    int n; long long s = 0;
    cin >> n;
    vector<int> a(n);
    for (auto &x : a) cin >> x;
    for (auto x : a) s = s + x;
    cout << s;
    return 0;
```

4.1.3 Định nghĩa và khai báo mảng hai chiều

Trong các ứng dụng toán, cấu trúc mảng hai chiều hay còn gọi là ma trận thường xuyên gặp ở nhiều trong thực tế, có thể liệt kê gồm:

- Biểu diễn đồ thị trong lý thuyết đồ thị.
- Ma trận ngẫu nhiên được sử dụng để tìm xích Markov với những trạng thái hữu hạn.
- Tìm nghiệm của các hệ phương trình tuyến tính.
- Và vô số ứng dụng khác.

Tương tự mảng một chiều, mảng hai chiều cũng được tiếp cận theo hai cách, một là khai báo tĩnh, hai là khai báo động. Ưu nhược điểm cũng tương tự như đã phân tích trên.

Khai báo tĩnh

Cú pháp: <TYPE> array2DName[MAXROW][MAXCOL];

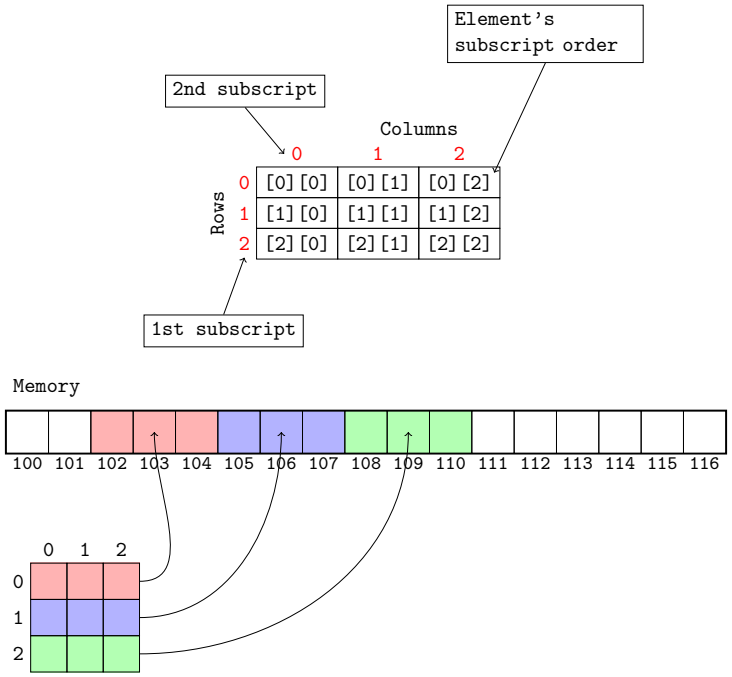
Trong đó:

- TYPE: Là các kiểu dữ liệu cơ bản, ví dụ như :int, long long, double.
- array2DName: Tên biến mảng hai chiều.
- MAXROW: Số phần tử tối đa tính theo dòng của mảng hai chiều cần khai báo, người ta còn gọi là chỉ số thứ nhất (1st subscript).

- MAXCOL: Số phần tử tối đa tính theo cột của mảng hai chiều cần khai báo, người ta còn gọi là chỉ số thứ hai (2nd subscript).

Xét ví dụ cho khai báo sau:char A[3][3];

Hình ảnh sau cho ta thấy các chỉ số, thuộc tính và cấu trúc bộ nhớ của mảng hai chiều vừa khai báo.



Hình 4.2: Mảng hai chiều và chỉ số

Khai báo động bằng vector

Cú pháp: vector< vector<TYPE> > v2D;

Chú ý: có ký tự trắng sau, trước cặp dấu ngoặc nhọn đầu tiên.

Xét ví dụ cho đoạn thuật toán nhân hai ma trận $C_{n\times p} = A_{n\times m} \times B_{m\times p}$ sau:

```
typedef long long int lli;
typedef vector<lli> vi;
typedef vector<vi> vvi;

...
vvi mul(vvi v1, vvi v2, int n, int p, int m) {
    vvi r(n, vi(m, 0));
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < m; j++) {
            for (int k = 0; k < p; k++) {
                r[i][j] = r[i][j] + v1[i][k] * v2[k][j];
```

```
        }
      }
    }
    return r;
}
```

4.1.4 Truy cập, xuất/nhập và duyệt mảng hai chiều

Mảng hai chiều cũng là cấu trúc tuyến tính, các phần tử đều được đánh chỉ mục cho nên dễ dàng truy cập các phần tử với thời gian $\mathcal{O}(1)$.

Cú pháp: array2DName[1Idx][2Idx];

Trong đó:

- array2DName: là tên mảng hai chiều do người lập trình đặt.
- 1Idx: là chỉ mục nhận giá trị từ 0 đến m tính theo dòng.
- 2Idx: là chỉ mục nhận giá trị từ 0 đến n tính theo cột.

Ví dụ 4.9. Biến đổi ma trận

Cho ma trận $A_{m\times n}$ gồm các số 0 và 1. Hãy biến đổi ma trận trên theo quy tắc nếu có phần tử $a_{i,j}$ nào bằng 1 thì tất cả các phần tử trên cùng dòng và cột của nó đều bằng 1.

Ví dụ: $\begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$ sẽ biến đổi thành: $\begin{bmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Input: Dòng đầu tiên chứa chứa số hai số nguyên m, n là kích thước của mảng thỏa $1 \leq m, n \leq 100$. m dòng tiếp theo mỗi dòng có n phần tử cột của ma trận.

Output: In ra ma trận sau khi biến đổi.

Sample Input 1

3 4

1 0 0 1

0 0 1 0

0 0 0 0

Sample Output 1

1 1 1 1

1 1 1 1

1 0 1 1

Lời giải.

```
#include <bits/stdc++.h>
```

```cpp
using namespace std;

int m,n,a[101][101];
bool r[101],c[101];

int main()
{
   //Nhập kích thước mảng.
   cin >> m >> n;
   //Nhập mảng và biến đổi dòng, cột nếu a[i][j]=1 theo yêu cầu.
   for(int i = 0;i < m;i++)
      for(int j = 0;j < n;j++){
         cin>>a[i][j];
         if(a[i][j] == 1)
         r[i] = 1,c[j] = 1;
   }
   //In mảng.
   for(int i = 0;i < m;i++){
      for(int j = 0;j < m;j++)
         if(r[i] == 1 || c[j] == 1)
            cout<<1<<" ";
         else cout<<0<<" ";
            cout<<endl;
   }
   return 0;
}
```

Nhận xét:

- Khai báo đơn giản tương tự mảng 1 chiều và khai báo toàn cục.
- Bạn thử chuyển khai báo tĩnh và địa phương xem nếu chưa khởi tạo thì ảnh hưởng gì đến bài toán.
- Kích thước dòng và cột của ma trận sẽ hạn chế vì số phần tử của ma trận = số dòng x số cột.

4.2 Bài tập chương 4

4.2.1 Bài tập về mảng một chiều

Bài toán 4.1. Đếm số chính phương trong dãy

Số chính phương biểu thị diện tích của một hình vuông có chiều dài cạnh bằng số tự nhiên. Cho dãy n số nguyên $A = \{a_1, a_2, \ldots, a_n\}$. Hãy lập trình đếm xem có bao nhiêu số chính phương có trong dãy A.

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 2.10^5$. Dòng thứ hai chứa các số $a_1, a_2, \ldots, a_n$ thỏa $1 \leq a_i \leq 10^9$.

Output: In ra kết quả cần đếm.

Sample Input 1

10

3 3 9 7 16 5 25 9 2 12

Sample Output 1

4

Link nộp bài: https://coder.husc.edu.vn/problem/pb0053

Bài toán 4.2. Đếm phần tử âm, dương, chẵn, lẻ

Cho dãy n số nguyên $A = \{a_1, a_2, \ldots, a_n\}$. Hãy lập trình đếm số các phần tử có giá trị: âm, dương, chẵn, lẻ xuất hiện trong dãy A.

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 2.10^5$. Dòng thứ hai chứa các số $a_1, a_2, \ldots, a_n$ thỏa $|a_i| \leq 10^9$.

Output: In ra các phần tử trên từng dòng, gồm:

- Số lượng số âm.

- Số lượng số dương.

- Số lượng số chẵn.

- Số lượng số lẻ.

Sample Input 1

10

3 -3 9 7 -12 -5 0 9 2 12

Sample Output 1

3

6

4

6

Link nộp bài: https://coder.husc.edu.vn/problem/pb0054

Bài toán 4.3. Tính tổng các phần tử trong dãy

Cho dãy n số nguyên $A = \{a_1, a_2, \ldots, a_n\}$. Hãy lập trình tính tổng các phần tử của dãy A.

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 2.10^5$. Dòng thứ hai chứa các số $a_1, a_2, \ldots, a_n$ thỏa $|a_i| \leq 10^9$.

Output: In ra tổng cần tính.

Sample Input 1

10

1 2 3 4 5 6 7 8 9 10

Sample Output 1

55

Link nộp bài: https://coder.husc.edu.vn/problem/pb0055

Bài toán 4.4. Tổng dãy số trừ phần tử max

Cho một dãy gồm n số nguyên $A = \{a_1, a_2, \ldots, a_n\}$. Hãy lập trình tính tổng các phần tử của dãy ngoại trừ các phần tử lớn nhất.

Input: Dòng đầu tiên chứa số nguyên dương n là số phần tử của dãy thỏa $n \leq 10^5$. Dòng thứ hai chứa các số $a_1, a_2, \ldots, a_n$ thỏa $|a_i| \leq 10^9$.

Output: In ra tổng cần tính.

Sample Input 1

5

5 9 3 4 6

Sample Output 1

18

Link nộp bài: https://coder.husc.edu.vn/problem/pb0056

Bài toán 4.5. Đếm số thỏa điều kiện

Cho một dãy gồm n số nguyên $A = \{a_1, a_2, \ldots, a_n\}$. Hãy lập trình đếm xem trong dãy có bao nhiêu số thỏa mãn một trong các yêu cầu sau:

- Nếu a_i là số chẵn.

- Nếu a_i nhỏ hơn 0 và chia hết cho 3.

Input: Dòng đầu tiên chứa số nguyên dương n là số phần tử của dãy thỏa $n \leq 10^5$. Dòng thứ hai chứa các số $a_1, a_2, \ldots, a_n$ thỏa $|a_i| \leq 10^9$.

Output: In ra kết quả cần đếm.

Sample Input 1

10

-5 -4 -3 -2 -1 0 1 2 3 4

Sample Output 1

6

Link nộp bài: https://coder.husc.edu.vn/problem/pb0057

Bài toán 4.6. Khoảng cách giá trị Min - Max

Cho dãy n số nguyên $A = \{a_1, a_2, \ldots, a_n\}$. Hãy lập trình in ra khoảng cách vị trí của phần tử lớn nhất đến vị trí phần tử nhỏ nhất của dãy trên.

Input: Dòng thứ nhất chứa số nguyên dương n thỏa $1 \leq n \leq 2.10^5$. Dòng thứ hai chứa n phần tử số nguyên a_i thỏa $|a_i| \leq 10^9$.

Output: In ra kết quả cần tìm.

Sample Input 1

7

1 2 3 7 1 2 5

Sample Output 1

3

Link nộp bài: https://coder.husc.edu.vn/problem/pb0058

Bài toán 4.7. Thứ hạng của các phần tử của dãy

Cho dãy số nguyên n phần tử $A = \{a_1, a_2, \ldots, a_n\}$. Hãy in ra thứ hạng của các phần tử theo chiều giảm dần, hạng bắt đầu từ 1. Hai phần tử bằng nhau thì chung một hạng.

Input: Dòng đầu tiên chứa số nguyên N thỏa $1 \leq N \leq 5000$. Dòng thứ hai chứa các n số nguyên a_i thỏa $1 \leq a_i \leq 10^9$.

Output: In ra hạng của lần lượt từng phần tử, mỗi hạng trên một dòng.

Sample Input 1

7

1 2 1 1 2 3 4

Sample Output 1

5

3

5

5

3

2

1

Link nộp bài: https://coder.husc.edu.vn/problem/pb0059

Bài toán 4.8. Xuất hiện lẻ lần

Cho dãy số nguyên $A = \{a_1, a_2, \ldots, a_n\}$ với n là số lẻ. Trong dãy này có một phần tử xuất hiện lẻ lần, các phần tử còn lại xuất hiện chẵn lần. Lập trình tìm phần tử đó.

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 2.10^5$. Dòng thứ hai chứa các n số nguyên a_i thỏa $0 \leq a_i \leq 10^9$.

Output: In ra phần tử có giá trị xuất hiện lẻ lần.

Sample Input 1

7

2 2 1 1 1 2 1

Sample Output 1

2

Link nộp bài: https://coder.husc.edu.vn/problem/pb0060

Bài toán 4.9. Trung bình cộng của dãy số

Cho dãy n số nguyên $A = \{a_1, a_2, \ldots, a_n\}$. Một phép toán thống kê cơ bản đặt ra là tính giá trị trung bình cộng của dãy trên. Hãy lập trình giải quyết nội dung trên.

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 2.10^5$. Dòng thứ hai chứa các n số nguyên a_i thỏa $|a_i| \leq 10^9$.

Output: In ra giá trị trung bình cộng của dãy với 2 chữ số sau dấu chấm thập phân.

Sample Input 1

3

1 2 3

Sample Output 1

2.00

Link nộp bài: https://coder.husc.edu.vn/problem/pb0061

Bài toán 4.10. Tổng điểm tốp đầu

Cho danh sách gồm n phần tử là các điểm thi của n sinh viên. Anh chị hãy lập trình tính tổng điểm của t người có điểm cao nhất.

Input: Dòng đầu tiên chứa hai số nguyên dương n, t thỏa $1 \leq t \leq n \leq 5000$. Dòng thứ hai chứa n số thực là điểm thi của các sinh viên thang điểm 10, có lấy lẻ một chữ số.

Output: In ra tổng cần tìm với 2 chữ số sau dấu chấm thập phân.

Sample Input 1

12 4

1 2 3 4 5 6 7 8.3 9 9 10 1

Sample Output 1

36.30

Link nộp bài: https://coder.husc.edu.vn/problem/pb0062

Bài toán 4.11. Liệt kê số siêu may mắn

Số siêu may mắn có giá trị trong từ 1 đến 100 gồm các số sau: $4, 7, 16, 28, 44, 47, 49, 64, 74$ và 77. Hãy đếm xem trong một mảng $A = \{a_1, a_2, \ldots, a_n\}$ có bao nhiêu số là số siêu may mắn như đề cập ở trên.

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 10^5$. Dòng thứ hai chứa n số nguyên a_i thỏa $a_i \leq 10^9$.

Output: In ra kết quả cần đếm.

Sample Input 1

10

1 2 3 4 5 6 7 8 9 10

Sample Output 1

2

Link nộp bài: https://coder.husc.edu.vn/problem/pb0063

Bài toán 4.12. Giá trị trung bình và trung vị

Cho dãy n số nguyên $a_1, a_2, \ldots, a_n$. Hãy lập trình thống kê các giá trị sau:

- Trung bình cộng của dãy số.

- Trung vị của dãy (nếu n lẻ thì trung vị là số ở giữa của dãy được sắp, nếu n chẵn thì trung vị là trung bình cộng của hai số giữa của dãy được sắp).

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 3000$. Dòng thứ hai chứa các số a_i thỏa $|a_i| \leq 10^6$.

Output: Dòng thứ nhất in giá trị trung bình với 6 chữ số sau dấu chấm thập phân. Dòng thứ hai in giá trị trung vị với 6 chữ số sau dấu chấm thập phân.

Sample Input 1

5

12 4 6 8 2

Sample Output 1

6.400000

6.000000

Link nộp bài: https://coder.husc.edu.vn/problem/pb0064

Bài toán 4.13. Tích hai vector

Cho hai vector $A = \{a_1, a_2, \ldots, a_n\}, B = \{b_1, b_2, \ldots, b_n\}$. Lập trình xác định xem tích hai vector trên có bằng 0 hay không?

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 10^5$. Dòng thứ hai chứa n số nguyên a_i là các phần tử của vector A thỏa $|a_i| \leq 100$. Dòng thứ ba chứa n số nguyên b_i là các phần tử của vector B thỏa $|b_i| \leq 100$.

Output: In ra **Yes** nếu thỏa và **No** nếu ngược lại.

Sample Input 1

2

-3 6

4 2

Sample Output 1

Yes

Link nộp bài: https://coder.husc.edu.vn/problem/pb0065

Bài toán 4.14. Sắp xếp dãy số

Cho dãy n số nguyên $A = \{a_1, a_2, \ldots, a_n\}$. Để sắp thứ tự tăng dần của các phần tử của dãy số trên **Thuận** tìm được rất nhiều thuật toán sắp có thể làm được công việc đó, bao gồm:

- Thuật toán sắp xếp nổi bọt (https://en.wikipedia.org/wiki/Bubble_sort).

- Thuật toán sắp xếp chèn (https://en.wikipedia.org/wiki/Insertion_sort).

- Thuật toán sắp xếp trộn (https://en.wikipedia.org/wiki/Merge_sort).

Và vô số thuật toán sắp xếp khác. Thuận cố gắng cài từng thuật toán một trong các thuật toán kể trên nhưng vẫn chưa **accept** được. Các bạn hãy lập trình giúp Thuận.

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 5000$. Dòng thứ hai chứa các n số nguyên a_i thỏa $|a_i| \leq 10^6$.

Output: In dãy được sắp theo thứ tự tăng dần.

Sample Input 1

5

3 1 2 -9 12

Sample Output 1

-9 1 2 3 12

Link nộp bài: https://coder.husc.edu.vn/problem/pb0066

Bài toán 4.15. Xếp hạng điểm số

Điểm số của các sinh viên từ hệ niên chế cần được chuyển sang hệ tín chỉ và cần được xếp hạng điểm bằng chữ như **D, D+, C, C+, B, B+, A** tùy theo thang điểm 10 theo quy tắc:

- **A** (8.5- 10) Giỏi.

- **B+** (8.0 - 8.4) Khá giỏi.

- **B** (7.0 - 7.9) Khá

- **C+** (6.5 - 6.9) Trung bình khá

- **C** (5.5 - 6,4) Trung bình.

- **D+** (5.0 - 5.4) Trung bình yếu.

- **D** (4.0 - 4.9) Yếu

Hãy lập trình xử lý yêu cầu trên.

Input: Dòng đầu tiên chứa một số nguyên dương n thỏa $1 \leq n \leq 5000$. Dòng thứ hai chứa n số thực là điểm thi của các sinh viên thang điểm 10, có lấy lẻ một chữ số.

Output: In ra danh sách với điểm xếp hạng mới tương ứng.

Sample Input 1

5

5 3 9 9 10

Sample Output 1

D+ D A A A

Link nộp bài: https://coder.husc.edu.vn/problem/pb0067

Bài toán 4.16. Tỷ số điểm thi

Thầy Bình tổ chức thi Competitive Programming cho sinh viên K42. Đề thi có n câu, mỗi câu tối đa 100 điểm. Hùng và Cường cá cược với nhau xem ai thắng bằng cách so điểm thi của các bài giải. Tuy nhiên không tính xem tổng điểm của tất cả các câu mà tính theo cách sau: Với mỗi từng câu một ai hơn điểm sẽ ghi 1 bàn, bằng điểm nhau không tính. Sau khi so hết tất cả các câu ai nhiều điểm hơn sẽ thắng.

Anh chị lập trình đếm xem tỷ số giữa Hùng và Cường là bao nhiêu?

Input: Dòng đầu tiên chứa số nguyên dương N là số phần tử của dãy thỏa $N \leq 100$. Dòng thứ hai chứa các số $a_1, a_2, \ldots, a_N$ là điểm thi các câu của Hùng. Dòng thứ ba chứa các số $b_1, b_2, \ldots, b_N$ là điểm thi các câu của Cường.

Output: In ra tỷ số điểm của Hùng và Cường.

Sample Input 1

5

7 96 12 48 53

7 90 40 50 16

Sample Output 1

2 2

Link nộp bài: https://coder.husc.edu.vn/problem/pb0068

Bài toán 4.17. Tổng nợ của công nhân

Bi viết một dãy số gồm n số là tiền lương của các công nhân, tuy nhiên lương lại có giá trị âm do các công nhân này tạm ứng trước lương (nhiều hơn số tiền được trả). Hãy lập trình tính tổng các giá trị âm ở trong dãy.

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 2.10^5$. Dòng thứ hai chứa các số $a_1, a_2, \ldots, a_n$ là tiền lương của các công nhân thỏa $|a_i| \leq 10^9$.

Output: In ra tổng nợ cần tính

Sample Input 1

10

3 -3 9 7 -12 -5 0 9 2 12

Sample Output 1

-20

Link nộp bài: https://coder.husc.edu.vn/problem/pb0069

Bài toán 4.18. Dãy răng cưa

Cho một dãy n số nguyên $A = \{a_1, a_2, \ldots, a_n\}$. Dãy răng cưa là dãy thỏa tính chất sau: $a_1 < a_2 > a_3 < a_4 \ldots$ hoặc $a_1 > a_2 < a_3 > a_4 \ldots$. Hãy lập trình giúp Bi xác định dãy số trên có phải là dãy răng cưa hay không?

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $3 \leq n \leq 10^5$. Dòng thứ hai chứa các số nguyên a_i thỏa $1 \leq a_i \leq 10^9$.

Output: In ra **Yes** nếu thỏa và **No** nếu ngược lại.

Sample Input 1

3

4 1 5

Sample Output 1

Yes

Link nộp bài: https://coder.husc.edu.vn/problem/pb0070

Bài toán 4.19. In dãy sau khi sắp thứ tự

Cho dãy số = $\{a_1, a_2, \ldots, a_n\}$. Hãy lập trình thực hiện các công việc sau:

- Sắp thứ tự tăng dần.

- In ra x phần tử đầu dãy và y phần tử cuối dãy vừa sắp thứ tự xong.

Input: Dòng đầu tiên chứa ba số nguyên dương n, x, y thỏa $2 \leq n \leq 10^3; x + y \leq n$. Dòng thứ hai chứa các số nguyên a_i thỏa $1 \leq a_i \leq 10^9$.

Output: In ra theo yêu cầu.

Sample Input 1

10 2 3

5 4 937 883 353 1 2 3 229 23

Sample Output 1

1 2 353 883 937

Link nộp bài: https://coder.husc.edu.vn/problem/pb0071

Bài toán 4.20. Kiểm tra mảng tăng dần

Cho dãy số $a = \{a_1, a_2, \ldots, a_n\}$. Hãy lập trình kiểm tra xem dãy số trên có tăng dần hay không?

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 10^5$. Dòng thứ hai chứa các số nguyên a_i thỏa $1 \leq a_i \leq 10^9$.

Output: In ra **Yes** nếu tăng dần và **No** nếu ngược lại.

Sample Input 1

10

5 4 937 883 353 1 2 3 229 23

Sample Output 1

No

Link nộp bài: https://coder.husc.edu.vn/problem/pb0072

Bài toán 4.21. Kiểm tra dãy đối xứng

Cho dãy số $A = \{a_1, a_2, \ldots, a_n\}$. Hãy lập trình kiểm tra xem dãy số trên có đối xứng hay không?

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 2.10^5$. Dòng thứ hai chứa các số nguyên a_i thỏa $1 \leq a_i \leq 10^9$.

Output: In ra **Yes** nếu đối xứng và **No** nếu ngược lại.

Sample Input 1

4

1 2 3 4

Sample Output 1

No

Link nộp bài: https://coder.husc.edu.vn/problem/pb0073

Bài toán 4.22. Sắp xếp chẵn, lẻ

Hãy sắp xếp dãy số gồm n phần tử $A = \{a_1, a_2, \ldots, a_n\}$ với yêu cầu là số chẵn sắp trước, số lẻ sắp sau và tăng dần.

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 3000$. Dòng thứ hai chứa các số nguyên a_i thỏa $1 \leq a_i \leq 10^9$.

Output: In ra dãy được sắp theo yêu cầu.

Sample Input 1

5

2 3 1 4 5

Sample Output 1

2 4 1 3 5

Link nộp bài: https://coder.husc.edu.vn/problem/pb0074

Bài toán 4.23. Tìm vị trí chèn trái

Cho một dãy số gồm n phần tử $A = \{a_1, a_2, \ldots, a_n\}$ được sắp thứ tự tăng dần. Hãy lập trình tìm vị trí thích hợp (tính từ trái sang) để chèn phần tử x vào dãy mà vẫn đảm bảo tính tăng dần.

Input: Dòng đầu tiên chứa hai số nguyên dương n, x thỏa $1 \leq n \leq 10^5; 1 \leq x \leq 10^9$. Dòng thứ hai chứa các số nguyên a_i thỏa $1 \leq a_i \leq 10^9$.

Output: In ra vị trí cần tìm, dãy đánh số thứ tự từ 1

Sample Input 1

10 3

1 1 2 2 3 3 3 5 6 9

Sample Output 1

5

Link nộp bài: https://coder.husc.edu.vn/problem/pb0075

Bài toán 4.24. Tìm vị trí chèn phải

Cho một dãy số gồm n phần tử $A = \{a_1, a_2, \ldots, a_n\}$ được sắp thứ tự tăng dần. Hãy lập trình tìm vị trí thích hợp (tính từ phải sang) để chèn phần tử x vào dãy mà vẫn đảm bảo tính tăng dần.

Input: Dòng đầu tiên chứa hai số nguyên dương n, x thỏa $1 \leq n \leq 10^5; 1 \leq x \leq 10^9$. Dòng thứ hai chứa các số nguyên a_i thỏa $1 \leq a_i \leq 10^9$.

Output: In ra vị trí cần tìm. Dãy đánh số thứ tự từ 1

Sample Input 1

10 4

1 1 2 2 3 3 3 5 6 9

Sample Output 1

8

Link nộp bài: https://coder.husc.edu.vn/problem/pb0076

Bài toán 4.25. Điền số Lucas

Trong lý thuyết số có rất nhiều dãy số có quan hệ rất đặc biệt, ví dụ như dãy Fibonacci, dãy số Armstrong. Hôm nay Bi đọc trên internet ở địa chỉ (http://oeis.org/) mới phát hiện ra một dãy số Lucas được định nghĩa như sau:

$$\mathrm{L_n} = \begin{cases} 2 & \text{nếu } n = 0 \\ 1 & \text{nếu } n = 1 \\ L_{n-1} + L_{n-2} & \text{nếu } n \geq 2 \end{cases}$$

Bi nhờ Anh chị lập trình bằng cách điền lần lượt các số $L_0, L_1, \ldots, L_n$ vào một dãy L gồm $n+1$ số Lucas ở trên. Sau đó in dãy ra màn hình.

Input: Dòng duy nhất chứa số nguyên dương n thỏa $1 \leq n \leq 80$.

Output: In dãy gồm $n+1$ số trên ra màn hình.

Sample Input 1

5

Sample Output 1

2 1 3 4 7 11

Link nộp bài: https://coder.husc.edu.vn/problem/pb0077

Bài toán 4.26. Đếm số bước lặp trên dãy

Cho một mảng số nguyên $A = \{a_1, a_2, \ldots, a_n\}$. Nếu giá trị các số của dãy đều chẵn thì Bi thực hiện thao tác chia các số trong dãy cho 2. Quá trình này sẽ lặp đi lặp lại cho đến khi trong dãy xuất hiện số lẻ. Hãy lập trình đếm xem Bi thực hiện bao nhiêu thao tác để dãy xuất hiện số lẻ.

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 100$. Dòng thứ hai chứa các số nguyên a_i thỏa $1 \leq a_i \leq 10^9$.

Output: In ra số bước lặp cần đếm.

Sample Input 1

3

8 12 40

Sample Output 1

2

Link nộp bài: https://coder.husc.edu.vn/problem/pb0078

4.2.2 Bài tập về mảng nhiều chiều

Bài toán 4.27. Tổng các phần tử trên dòng của ma trận

Cho một ma trận $A_{m,n}$ chứa các số nguyên, đọc là ma trận A gồm m dòng, n cột. Hãy lập trình in ra tổng các phần tử của từng dòng của ma trận A. Ví dụ ma trận $m = 3, n = 3$ với dữ liệu sau:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Ta có tổng các phần tử dòng đầu tiên bằng 6.

Input: Dòng đầu tiên chứa hai số nguyên dương m, n thỏa $1 \leq m, n \leq 1000$. m dòng tiếp theo, mỗi dòng chứa n số $a_{i,j}$ của ma trận thỏa $1 \leq a_{i,j} \leq 10^6$.

Output: In dãy m phần tử là tổng lần lượt các dòng của ma trận.

Sample Input 1

3 3

1 2 3

4 5 6

7 8 9

Sample Output 1

6 15 24

Link nộp bài: https://coder.husc.edu.vn/problem/pb0079

Bài toán 4.28. Tổng các phần tử trên cột của ma trận

Cho một ma trận $A_{m,n}$ chứa các số nguyên, đọc là ma trận A gồm m dòng, n cột. Hãy lập trình in ra tổng các phần tử của từng cột của ma trận A. Ví dụ ma trận $m = 3, n = 3$ với dữ liệu sau:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

Ta có tổng các phần tử cột đầu tiên bằng 12.

Input: Dòng đầu tiên chứa hai số nguyên dương m, n thỏa $1 \leq m, n \leq 1000$. m dòng tiếp theo, mỗi dòng chứa n số $a_{i,j}$ của ma trận thỏa $1 \leq a_{i,j} \leq 10^6$.

Output: In dãy m phần tử là tổng lần lượt các cột của ma trận.

Sample Input 1

3 3

1 2 3

4 5 6

7 8 9

Sample Output 1

12 15 18

Link nộp bài: https://coder.husc.edu.vn/problem/pb0080

Bài toán 4.29. Ma trận chuyển vị

Nếu A là một ma trận có kích thước $m \times n$ với các giá trị là $a_{i,j}$ tại hàng i, cột j, thì ma trận chuyển vị $B = A^T$ sẽ là ma trận có kích thước $n \times m$ với các giá trị $b_{i,j} = a_{j,i}$. Hãy lập trình in ra ma trận chuyển vị của một ma trận A cho trước.

Input: Dòng đầu tiên chứa hai số nguyên dương m, n thỏa $1 \leq m, n \leq 100$. m dòng tiếp theo, mỗi dòng chứa n số $a_{i,j}$ của ma trận thỏa $|a_{i,j}| \leq 10^9$.

Output: In ra ma trận chuyển vị của A.

Sample Input 1

3 4

13 9 7 15

8 7 4 6

6 4 0 3

Sample Output 1

13 8 6

9 7 4

7 4 0

15 6 3

Link nộp bài: https://coder.husc.edu.vn/problem/pb0081

Bài toán 4.30. Điểm yên ngựa

Cho mảng hai chiều A có kích thước $M \times N$ số nguyên. Phần tử $A_{i,j}$ được gọi là phần tử **yên ngựa** nếu nó là phần tử nhỏ nhất trong hàng i và đồng thời là phần tử lớn nhất trong cột j. Bạn hãy viết chương trình tìm điểm yên ngựa của mảng A.

Input: Dòng dầu tiên gồm hai số M, N thỏa $0 \leq M, N \leq 100$. M dòng tiếp theo mỗi dòng gồm có N số nguyên của mảng A.

Output: In vị trí của các phần tử **yên ngựa** nếu có hoặc in thông báo **Khong co phan tu yen ngua**. Mẫu in như ví dụ mẫu sau:

Sample Input 1

3 3

15 3 9

55 4 6

76 1 2

Sample Output 1

Cac phan tu yen ngua la:

(2,2);

Sample Input 2

3 4

15 10 8 8

55 4 6 2

76 9 12 8

Sample Output 2

Cac phan tu yen ngua la:

(1,4); (3,4);

Link nộp bài: https://coder.husc.edu.vn/problem/pb0082

Bài toán 4.31. Tổng hai chéo ma trận bản 1

Cho ma trận vuông kích thước n. Người ta đưa ra hai khái niệm là:

- Đường chéo chính là các phần tử mà chỉ số $i = j$.

- Đường chéo phụ là các phần tử mà chỉ số $i = n - 1 - j$.

Hãy lập trình tính tổng các phần tử trên hai đường chéo của ma trận.

Input: Dòng đầu tiên chứa số nguyên n là kích thước ma trận thỏa $1 \leq n \leq 500$. n dòng tiếp theo, mỗi dòng chứa n số nguyên $a_{i,j}$ của ma trận thỏa $1 \leq a_{i,j} \leq 10^6$.

Output: In ra tổng cần tính.

Sample Input 1

3

1 2 3

4 5 6

7 8 9

Sample Output 1

25

Link nộp bài: https://coder.husc.edu.vn/problem/pb0083

Bài toán 4.32. Các phần tử lớn nhất trong ma trận

Cho một ma trận A gồm m dòng và n cột. Các dòng được đánh số từ 1 đến m, các cột được đánh số từ 1 đến n. Phần tử ở hàng $i(1 \leq i \leq m)$ và cột $j(1 \leq j \leq n)$ được ký hiệu là $a_{i,j}$. Hãy lập trình:

- In ra các phần tử lớn nhất trên từng hàng của ma trận.

- In ra các phần tử lớn nhất trên từng cột của ma trận.

Input: Dòng đầu tiên chứa hai số nguyên m, n thỏa $1 \leq m, n \leq 100$. m dòng tiếp theo, mỗi dòng chứa n số nguyên $a_{i,j}$ thỏa $|a_{i,j}| \leq 10^9$.

Output: Dòng đầu tiên in các giá trị lớn nhất của từng hàng của ma trận. Dòng thứ hai in các giá trị lớn nhất của từng cột của ma trận.

Sample Input 1

4 5

4 8 8 9 4

8 5 2 4 0

2 3 3 8 7

6 6 5 0 4

Sample Output 1

9 8 8 6

8 8 8 9 7

Link nộp bài: https://coder.husc.edu.vn/problem/pb0084

Bài toán 4.33. Nhân ma trận

Hãy lập trình in ra ma trận tích của hai ma trận $A_{m,n} \times B_{n,p}$. Chú ý rằng bài toán chỉ thực hiện được khi số cột (n) của ma trận A bằng số hàng (n) của ma trận B và phép toán này không có tính giao hoán.

Input: Dòng đầu tiên gồm ba số nguyên m, n, p thỏa $1 \leq m, n, p \leq 100$. m dòng tiếp theo, mỗi dòng chứa n số nguyên $a_{i,j}$ thỏa $|a_{i,j}| \leq 10^9$ mô tả các hàng của ma trận A. n dòng tiếp theo, mỗi dòng chứa p số nguyên $b_{i,j}$ thỏa $|b_{i,j}| \leq 10^9$ mô tả các hàng của ma trận B.

Output: In ra ma trận tích của hai ma trận trên.

Sample Input 1

1 3 4

3 4 2

13 9 7 15

8 7 4 6

6 4 0 3

Sample Output 1

83 63 37 75

Link nộp bài: https://coder.husc.edu.vn/problem/pb0085

Bài toán 4.34. Tổng hai ma trận

Cho hai ma trận cùng kích thước $A_{m,n}, B_{m,n}$ chứa các số nguyên. Hãy lập trình in ra ma trận tổng của hai ma trận trên. Tổng hai ma trận trên được định nghĩa như sau: $c_{i,j} = a_{i,j} + b_{i,j}$ với $1 \leq i \leq m; 1 \leq j \leq n$.

Input: Dòng đầu tiên chứa hai số nguyên dương m, n thỏa $1 \leq m, n \leq 200$. m dòng tiếp theo, mỗi dòng chứa n số $a_{i,j}$ của ma trận A thỏa $0 \leq a_{i,j} \leq 1000$. m dòng kế tiếp, mỗi dòng chứa n số $b_{i,j}$ của ma trận B thỏa $0 \leq b_{i,j} \leq 1000$.

Output: In ma trận tổng C.

Sample Input 1

3 3

1 2 3

4 5 6

7 8 9

9 8 7

6 5 4

3 2 1

Sample Output 1

10 10 10

10 10 10

10 10 10

Link nộp bài: https://coder.husc.edu.vn/problem/pb0086

Bài toán 4.35. Ma trận bằng nhau

Cho hai ma trận cùng kích thước $A_{m,n}, B_{m,n}$ chứa các số nguyên. Hãy lập trình kiểm tra xem hai ma trận trên có bằng nhau hay không?

Input: Dòng đầu tiên chứa hai số nguyên dương m, n thỏa $1 \leq m, n \leq 1000$. m dòng tiếp theo, mỗi dòng chứa n số $a_{i,j}$ của ma trận A thỏa $0 \leq a_{i,j} \leq 10^6$. m dòng kế tiếp, mỗi dòng chứa n số $b_{i,j}$ của ma trận B thỏa $0 \leq b_{i,j} \leq 10^6$.

Output: In ra **Yes** nếu bằng nhau và ngược lại in **No**.

Sample Input 1 3 3

1 2 3

4 5 6

7 8 9

1 2 3

4 5 6

7 8 9

Sample Output 1

Yes

Link nộp bài: https://coder.husc.edu.vn/problem/pb0087

Bài toán 4.36. Ma trận đơn vị

Ma trận đơn vị là một ma trận vuông đặc biệt có các phần tử trên đường chéo chính bằng 1 và các phần tử khác bằng 0. Cho ma trận $A_{n,n}$ chứa các số nguyên. Hãy lập trình kiểm tra xem ma trận A có phải là ma trận đơn vị hay không?

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 500$. n dòng tiếp theo, mỗi dòng chứa n số $a_{i,j}$ của ma trận A thỏa $1 \leq a_{i,j} \leq 10^6$.

Output: In ra **Yes** nếu là ma trận đơn vị và ngược lại in **No**.

Sample Input 1

3

1 0 1

0 1 0

0 0 1

Sample Output 1

No

Link nộp bài: https://coder.husc.edu.vn/problem/pb0088

Bài toán 4.37. Ma trận thưa

Ma trận thưa là một ma trận có số phần tử 0 xuất hiện nhiều trong ma trận. Cụ thể, gọi T là tổng số các phần tử 0, ma trận $A_{m,n}$ là ma trận thưa khi $T \geq \frac{m \times n}{2}$.

Hãy lập trình kiểm tra xem ma trận A có phải là ma trận thưa hay không?

Input: Dòng đầu tiên chứa hai số nguyên dương m, n thỏa $1 \leq m, n \leq 500$. m dòng tiếp theo, mỗi dòng chứa n số $a_{i,j}$ của ma trận A thỏa $1 \leq a_{i,j} \leq 10^6$.

Output: In ra **Yes** nếu là ma trận thưa và ngược lại in **No**.

Sample Input 1

3 3

1 0 1

0 1 0

0 0 1

Sample Output 1

Yes

Link nộp bài: https://coder.husc.edu.vn/problem/pb0089

Chương 5

Lập trình Hàm

5.1 Tóm tắt lý thuyết

Ngôn ngữ lập trình bậc cao đưa ra khái niệm hàm (function) được sử dụng trong lập trình dưới cái nhìn của lập trình có cấu trúc. Việc xây dựng chương trình được thiết kế theo hai hướng là từ dưới lên (bottom up) theo hướng chức năng (module) và thiết kế theo hướng từ trên xuống (top down) theo kiểu đối tượng. Và cho dù theo hướng phát triển nào thì hàm (function) đều đóng vai trò quan trọng trong việc xây dựng chương trình.

Công thức của chương trình là: "Chương trình = Cấu trúc dữ liệu + Giải thuật", trong đó giải thuật (thuật toán) thường được xây dựng bằng một hàm.

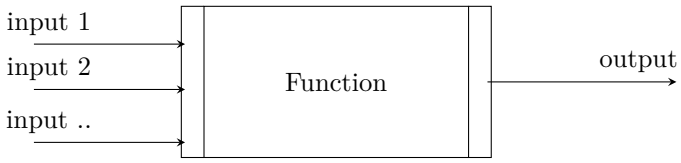
Ngoài ra, còn có công thức: "Đối tượng (object) = Dữ liệu + Hành vi", trong đó hành vi (phương thức, phép toán,..) thường được xây dựng bằng một hàm.

Tầm quan trọng của hàm chính là tính linh hoạt trong lắp ghép, dễ viết, dễ sửa, dễ tối ưu bằng thuật toán. Do vậy người ta hay đồng nhất với khái niệm module (dùng trong công nghệ lắp ráp công nghiệp). Về cơ bản hàm có cấu tạo như một chương trình nhưng mức độ nhỏ hơn, có tính độc lập cao nên còn được gọi là chương trình con (SubProgram).

C++ đưa ra các khái niệm về hàm như sau:

- Hàm dựng sẵn (build-in) được cung cấp bởi các thư viện, ví dụ #include<cmath.h> cho ta tập các hàm toán học xây dựng sẵn.
- Hàm phương thức (methode) gắn liền với đối tượng sử dụng thông qua các lớp (class) xây dựng trước.
- Hàm do người dùng định nghĩa.

Chương này ta tìm hiểu cách thức xây dựng và sử dụng hàm do mình tạo ra.



Hình 5.1: Cấu trúc của một hàm

5.1.1 Định nghĩa hàm

Hàm như một chương trình con độc lập, một hàm được xây dựng chặt chẽ nếu xác định rõ đầu vào (input), đầu ra (output) của hàm. Theo hình vẽ để định nghĩa hàm chính xác, ta cần xác định các thông tin sau:

- Tên hàm, định danh quan trọng cho việc gọi hàm.
- Đầu vào dữ liệu của hàm, đây là phần tử quan trọng gọi là tham số (parameter) của hàm.
- Đầu ra dữ liệu của hàm, thường trả về duy nhất một giá trị thông qua lệnh return.
- Đầu vào/ra cho hàm là tự chọn nghĩa là có thể rỗng (void).
- Thuật toán chính xác, hiệu quả cho việc cài đặt hàm.
- Sử dụng các biến địa phương, toàn cục phù hợp.

Cú pháp:

```
<TYPE> functionName( parameter_list ) {
   statement(s);
   return value;
}
```

Trong đó:

- TYPE: là kiểu dữ liệu trả về của hàm, có thể làn số nguyên (int, long long), số thực (double), xâu ký tự (string), bản ghi (struct), con trỏ (pointer),..
- functionName: là tên hàm do người lập trình đặt quy tắc tương tự như quy tắc đặt tên biến, hàm.
- args_list: là danh sách các đối số, đó chính là các đầu vào (input) của hàm. Cú pháp của danh sách đối số là TYPE agr1,...
- return value: chính là lệnh trả về kết quả (output) cuối cùng của hàm.

Ví dụ 5.1. Hàm xác định số nguyên tố

Một số tự nhiên $p (p > 1)$ là số nguyên tố nếu p có đúng hai ước số là 1 và p. Ví dụ các số nguyên tố: 2, 3, 5, 7, 11, 13, 17, 19, 23. Hãy viết hàm xác định xem một số nguyên n cho trước có phải là số nguyên tố hay không?

Lời giải 1. Thuật toán ngây thơ (naïve) bằng cách đếm xem có bao nhiêu ước số, nếu có chính xác 2 ước số thì đó là số nguyên tố.

```
bool isPrime(int n) {
   if (n<=1) return false;
   else{
      int c=0;
```

```
5       for (int i = 1; i <= n; i++)
6         if (n % i == 0) c++;
7       if (c==2) return true;
8       else return false;
9     }
10  }
```

Lời giải 2. Thuật toán ngây thơ (naïve) bằng cách chỉ cần có một ước số ngoài 1 và chính nó thì trả về false bằng lệnh return.

```
1  bool isPrime(int n) {
2    for (int i = 2; i <= n - 1; i++)
3      if (n % i == 0) return false;
4    return n > 1;
5  }
```

Nhận xét: câu lệnh return rất uy lực khi nó vừa đảm nhận chức năng trả về giá trị cuối cùng cho hàm và còn có khả năng thoát ra khỏi vòng lặp một cách tinh tế.

Các số nguyên lớn hơn 1 không phải là số nguyên tố thì gọi là hợp số. Nếu $n > 1$ là hợp số thì ta luôn có thể tách $n = i_1 \times i_2$ mà $2 \leq i_1 \leq i_2 \leq n - 1$. Vì $i_1 \times i_1 \leq i_1 \times i_2 = n$ nên $i_1 \leq \sqrt{n}$.

Do đó việc kiểm tra ước số từ 2 đến $n - 1$ là không cần thiết mà chỉ cần kiểm tra từ 2 đến $\sqrt{n}$.

Lời giải 3. Thuật toán tối ưu với độ phức tạp $\mathcal{O}(\sqrt{n})$ như sau:

```
1  bool isPrime(int n) {
2    for (int i = 2; i*i < n; i++)
3      if (n % i == 0) return false;
4    return n > 1;
5  }
```

Do số chẵn lớn hơn 2 không là số nguyên tố, như vậy chỉ cần kiểm tra i là số lẻ và nhỏ hơn hoặc bằng $\sqrt{n}$.

Lời giải 4. Thuật toán tối ưu với độ phức tạp $\mathcal{O}(\sqrt{n})$ như sau:

```
1  bool isPrime(int n) {
2    if ((n>2) and (n%2==0)) return false;
3    else{
4      for (int i = 3; i*i < n; i+=2)
5        if (n % i == 0) return false;
6      return n > 1;
7    }
8  }
```

Với ví dụ nhỏ thông qua các phương án giải ta thấy tính hiệu quả của viết hàm trong lập trình.

Ví dụ 5.2. Hàm sắp xếp dãy số bằng thuật toán Buble sort

Cho dãy gồm n số nguyên $a_1, a_2, \ldots, a_n$. Hãy viết hàm sắp xếp dãy số trên bằng thuật toán Buble sort (http://en.wikipedia.org/wiki/Bubble_sort)

Lời giải 1. Dựa theo nội dung giả mã của thuật toán ta có thể dễ dàng cài đặt bản đầu tiên cho thuật toán trên như sau:

```
void BubbleSort(int arr[], int length){
  int n = length;
  while(n > 0){
    int newn = 0;
    for(int i = 1; i < n; i++)
      if (arr[i-1]> arr[i]){
        std::swap(arr[i-1], arr[i]);
        newn = i;
      }
    n = newn;
  }
}
//Lời gọi trong hàm main()
int A[] = {1, 2, 5, 7, 6, 1, 3, 2, 9, 15, 14, 20};
int n=12;
BubbleSort(A, n);
```

Cải tiến: Một option được đưa vào đó là có thể sắp theo thứ tự tăng (ascending) hoặc giảm(descending), để giải quyết người ta sử dụng con trỏ trỏ đến hàm thứ tự sắp như một tham số trong hàm BubbleSort còn được gọi là **Con trỏ hàm**. Sau đó tùy theo yêu cầu sắp xếp mà truyền vào hàm sắp tương ứng.

Lời giải 2. Bản cải tiến với tham số sắp theo thứ tự tăng hoặc giảm.

```
void BubbleSort(int  arr[], int length, int (*pComparison)(int, int)){
  int n = length;
  while(n > 0){
    int newn = 0;
    for(int i = 1; i < n; i++)
      if (pComparison(arr[i-1], arr[i])>0){
        std::swap(arr[i-1], arr[i]);
        newn = i;
      }
    n = newn;
  }
}
//Hàm sắp xếp tăng dần.
int Ascending(int a, int b){
  return a > b;
}
//hàm sắp xếp giảm dần.
```

```
int Descending(int a, int b) {
   return b > a;
}
//Lời gọi trong hàm main().
int A[] = {1, 2, 5, 7, 6, 1, 3, 2, 9, 15, 14, 20};
int n=12;
BubbleSort(A, n, Ascending); // nếu sắp tăng
BubbleSort(A, n, Descending);// nếu sắp giảm
```

Nhận xét: Một đối số được đưa vào cho hàm là: int (*pComparison) (int, int) có kiểu int là địa chỉ của hàm được truyền vào, như vậy khi viết hàm truyền vào (như Accending(), Descending()) thì tham số hai hàm này phải quan hệ $1-1$ với đối số của *pComparison. Bản cài đặt hàm BubbleSort cải tiến chỉ thay việc so sánh if (arr[i-1]> arr[i]) bằng if(pComparison(arr[i-1], arr[i])>0)

5.1.2 Truyền tham số cho hàm

Xét ví dụ đoạn chương trình sau:

```
void foo(int agrs1, int agrs2) //agrs1, agrs2 gọi là hai đối số của hàm.
{
   cout << agrs1 << " " << agrs2;
}
int main()
{
   int x=1;
   foo(x, 2); //x và 2 là hai giá trị thực, tham gia vào việc gọi hàm và gọi là tham số thực.return 0;
```

Quan hệ giữa "**Đối số (args)- Tham số (parameter)**" là quan hệ 1-1 trên phương diện thứ tự, kiểu dữ liệu.

Đối số là hình thức mô tả đầu vào cho hàm

Tham số là giá trị thực sự (có thể là biến, hằng số, con trỏ hàm) tham gia vào việc gọi hàm.

Vậy truyền tham số thực sự vào cho hàm thông qua đối số (input) của hàm hoạt động như thế nào, chúng ta tìm hiểu qua các nội dung sau.

Truyền tham số thực bằng giá trị

Nguyên tắc: Tham số thực sự khi truyền vào hàm thông qua đối số sẽ không bị thay đổi dữ liệu bằng cách sao chép dữ liệu ra vùng nhớ mới và truyền vào cho hàm.

Xét ví dụ đoạn chương trình sau:

```
#include <iostream>
using namespace std;
void process(int value) {
   cout << "Value passed into function: " << value << endl;
```

```
    value = 10;
    cout << "Value before leaving function: " << value << endl;
}
int main() {
    int someValue = 7;
    cout << "Value before function call: " << someValue << endl;
    process(someValue);
    cout << "Value after function call: " << someValue << endl;
    return 0;
}
```

Khi chạy chương trình sẽ in ra kết quả như sau:

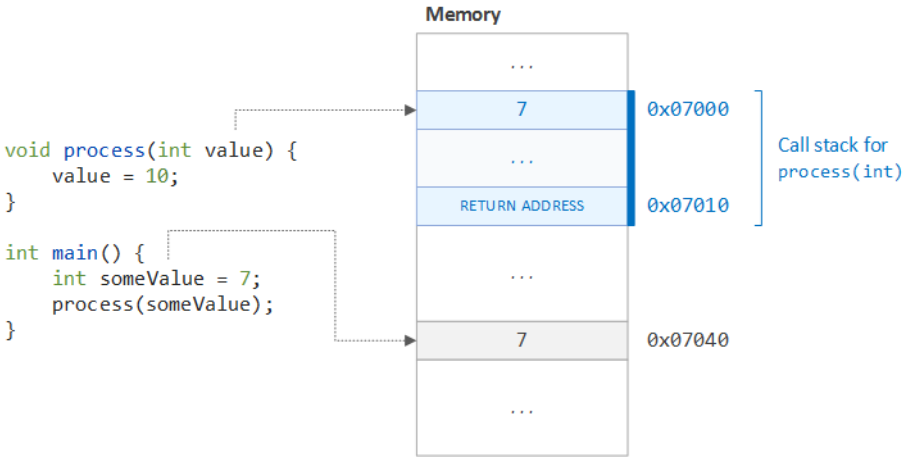
Value before function call: 7

Value passed into function: 7

Value before leaving function: 10

Value after function call: 7

Hình ảnh minh họa như sau:



Hình 5.2: Truyền bằng tham số trị

Truyền tham số thực bằng tham chiếu

Nguyên tắc: Tham số thực sự khi truyền vào hàm thông qua đối số bằng cách đưa địa chỉ nhớ của mình (thông qua phép toán &) cho hàm. Như vậy, các tác động cập nhật có thay đổi dữ liệu sẽ thay đổi trực tiếp trên tham số thực sự truyền vào.

Xét ví dụ đoạn chương trình sau:

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

```cpp
void process(int& value) {
    cout << "Value passed into function: " << value << endl;
    value = 10;
    cout << "Value before leaving function: " << value << endl;
}
int main() {
    int someValue = 7;
    cout << "Value before function call: " << someValue << endl;
    process(someValue);
    cout << "Value after function call: " << someValue << endl;
    return 0;
}
```

Khi chạy chương trình sẽ in ra kết quả như sau:

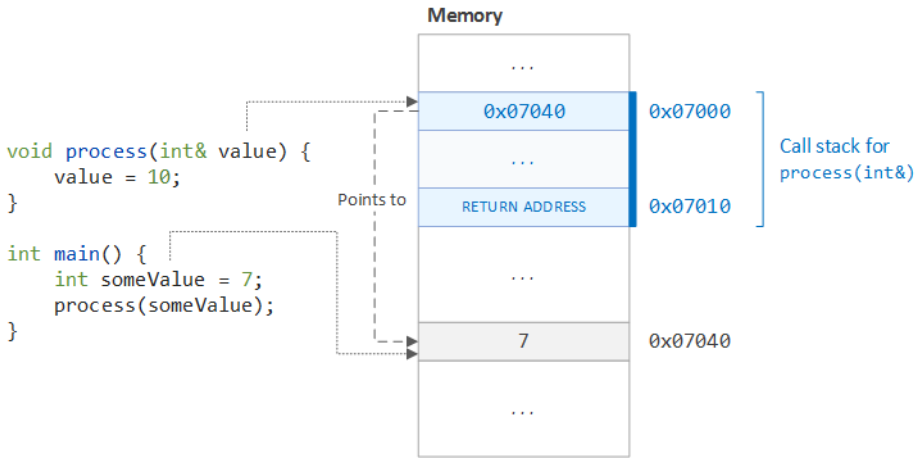
Value before function call: 7

Value passed into function: 7

Value before leaving function: 10

Value after function call: 10

Hình ảnh minh họa như sau:



Hình 5.3: Truyền bằng tham số chiếu

Kết luận:

- Truyền tham trị khi bạn không muốn giá trị của biến bị thay đổi.
- Truyền tham chiếu khi bạn muốn hàm thay đổi giá trị biến của bạn.
- Truyền tham chiếu sẽ tiết kiệm bộ nhớ hơn, vì nó không phải tạo ra 1 bản sao.
- Nếu bạn dùng tham chiếu, bạn cũng không muốn hàm thay đổi giá trị của bạn thì bạn cần thêm từ khóa const vào tham số đó. Ví dụ: int f(const int &x).
- Nếu tham số là mảng thì nó được truyền theo hình thức tham chiếu.

5.1.3 Hàm đệ quy

Đệ quy là một khái niệm được sử dụng trong nhiều lĩnh vực khác nhau như Toán học, Sinh học, Ngôn ngữ học, nghệ thuật. Trong lập trình máy tính, đệ quy (Recursive) là một chiến thuật giải quyết vấn đề một cách mạnh mẽ, tổng quát cho phép ta thiết kế thuật toán ngắn gọn, đơn giản và cải tiến được tốc độ tính toán cho các bài toán.

Nhận dạng đệ quy

Một thực thể (Entity) hoặc một khái niệm (Concept) được gọi là đệ quy khi nó được định nghĩa bởi chính nó. Một cách nói khác là một khái niệm X được gọi là định nghĩa đệ quy nếu trong định nghĩa X có sử dụng ngay chính khái niệm X. Xét hai ví dụ trong toán học sau:

Số n gọi là **số tự nhiên** khi:

- 0 là số tự nhiên,

- n là số tự nhiên nếu $n-1$ là số tự nhiên.

Số n **giai thừa**, ký hiệu $n!$ khi:

- $0! = 1$,

- Nếu $n > 0$ thì $n! = (n-1)! \times n$.

Trong tự nhiên ta có thể thấy sự đệ quy xuất hiện khắp mọi nơi, ví dụ: cấu trúc phân nhánh của một cây, sự phân nhánh của một dòng sông, cấu trúc sắp xếp của một bông hoa, búp bê Nga, sự fractal trong đồ họa ứng dụng nhiều trong nghệ thuật.

Trong toán học người ta sử dụng tính chất tổng quát hóa để định nghĩa các khái niệm nên xuất hiện rất nhiều khái niệm đệ quy. Ví dụ: Số Fibonacci, Số n giai thừa, công thức tính tổ hợp,...

Trong lập trình máy tính với lĩnh vực cấu trúc dữ liệu (Structure Data Type) là cách thức lưu trữ và tìm kiếm dữ liệu, người ta sử dụng kỹ thuật định nghĩa đệ quy để xử lý lưu trữ và tìm kiếm dữ liệu. Ví dụ: Danh sách liên kết, cây tìm kiếm nhị phân, phép duyệt cây cho các thuật toán tìm kiếm trên các cấu trúc lưu trữ chỉ mục.

Lập trình đệ quy

Ngôn ngữ lập trình bậc cao cung cấp cho ta cơ chế định nghĩa đệ quy đó chính là lập trình hàm (Function Programming). Hàm chính là một chương trình con (SubProgram) có cấu trúc như một chương trình chính (main) dùng để giải quyết một bài toán cụ thể nào đó. Vậy sự định nghĩa đó cũng có thể định nghĩa bằng đệ quy.

Một hàm đươc gọi là đệ quy nếu trong quá trình thực hiện nó có lời gọi (Call Function) đến chính nó.

Xét ví dụ đoạn mã sau:

```
1 long long fact(int n){
2   if (n==0) return 1;
3   else return fact(n-1) * n;//Lời gọi đệ quy: fact(n-1).
```

```
4 }
```

Cái quan trọng nhất của việc lập trình đệ quy chính là định nghĩa được bài toán bằng công thức đệ quy, bao gồm:

- Điểm cơ sở (suy biến của công thức).

- Công thức đệ quy tổng quát.

Ví dụ thuật toán sau:

```
int Collatz(int n) {
  if (n == 1) return 1;   //Điểm cơ sở
  else                    //Công thức đệ quy tổng quát.
    if (n%2 == 0) return 1 + Collatz(n/2);
    else return 1 + Collatz(3*n + 1);
}
```

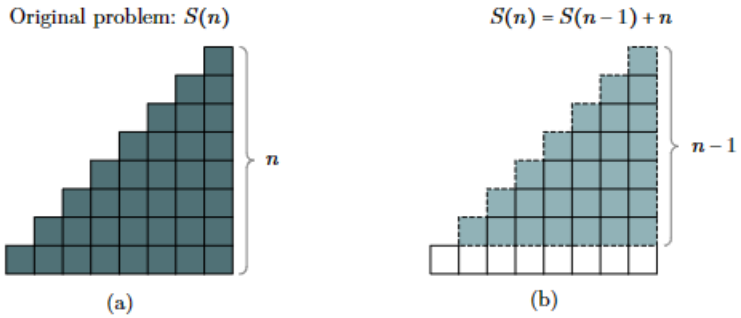
Ngoài ra, điểm ưu việt nhất chính là cách phân rã bài toán con với tham số như thế nào chính là vấn đề mấu chốt của lập trình đệ quy. Xét bài toán cụ thể sau:

Ví dụ 5.3. Tính $S = 1 + 2 + \ldots + n$

Viết hàm đệ quy tính tổng S với $n, n \leq 10^7$ nguyên dương cho trước.

Lời giải:

Bài toán được tiếp cận đệ quy bằng nhiều cách mô tả như hình vẽ sau:



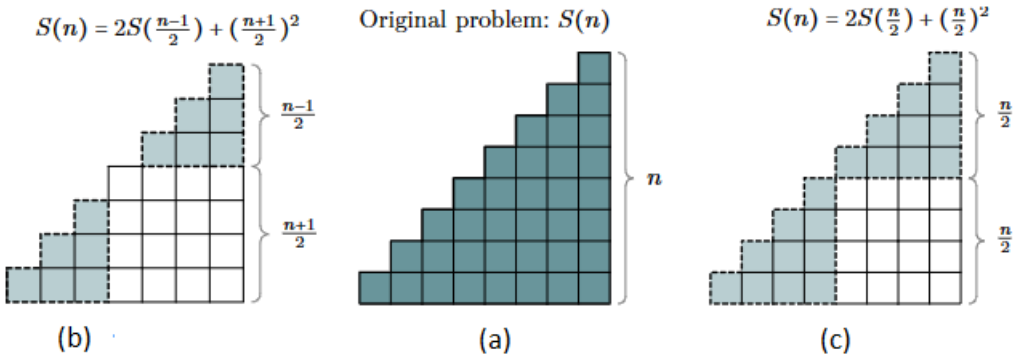
Hình 5.4: Tính tổng $S = 1 + 2 + \ldots + n$

Mã nguồn đệ quy cho cách phân rã thứ nhất - Hình 5.4.b

```
long long sum1(long long n){
  if (n==1) return 1;
  else return sum1(n-1) + n;//Lời gọi đệ quy.
}
```

Mã nguồn đệ quy cho cách phân rã thứ hai - Hình 5.5.b, 5.5.c

```
long long sqr(long long x){
  return x*x*1LL;
```



Hình 5.5: Tính tổng $S = 1 + 2 + \ldots + n$

```
}
long long sum(long long  n){
   if (n==1) return 1LL;
   else
      if (n%2==0)
         return 2 * sum (n/2) + sqr(n/2);
      else
         return 2 * sum((n-1)/2) + sqr((n+1)/2);
}
```

Có thể nộp bài ở link sau để chứng thực: https://coder.husc.edu.vn/problem/pb0040.

Và với ví dụ về cách phân rã như trên ta có khái niệm lập trình **đệ quy chia để trị**, một cách tối ưu thuật toán rất hữu hiệu. Có thể thấy qua các thuật toán sau:

- Thuật toán tìm kiếm nhị phân (Binary Search).
- Thuật toán sắp xếp nhanh (Quick Sort).
- Thuật toán sắp xếp trộng (Merge Sort).
- ...

5.2 Bài tập chương 5

Bài toán 5.1. Số tuyệt vời

Hôm nay **Bi** học về lập trình hàm (function) nên được thầy giáo giao nhiệm vụ viết hai hàm sau:

- Hàm đổi một số nguyên n sang dãy nhị phân.

- Hàm kiểm tra xem một dãy nhị phân có đối xứng hay không?

Bi thấy khó quá nên nhờ các anh chị giúp, nhớ viết dưới dạng hàm nghe, kẻo thầy cô không đồng ý. Để sau khi có các hàm trên **Bi** ráp vào giải bài toán kiểm tra xem một số nguyên n có phải là **số tuyệt vời** hay không, số tuyệt vời là số mà thỏa:

- Số lẻ.

- Biểu diễn nhị phân của nó là đối xứng.

Input: Dòng duy nhất chứa số nguyên n thỏa $0 \leq n \leq 10^9$.

Output: In ra **YES** nếu thỏa và **NO** nếu ngược lại.

Sample Input 1

3

Sample Output 1

YES

Link nộp bài: https://coder.husc.edu.vn/problem/pb0090

Bài toán 5.2. Số Armstrong

Hãy viết hàm kiểm tra xem một số nguyên dương n có phải là số **Armstrong** hay không?

Số **Armstrong** là một số tự nhiên có n chữ số nếu thỏa mãn điều kiện: tổng các lũy thừa bậc n của các chữ số của nó bằng chính nó. Ví dụ, số 153 là số **Armstrong** vì $153 = 1^3 + 5^3 + 3^3$

Áp dụng: Cho dãy n số nguyên $A = \{a_1, a_2, \ldots, a_n\}$. Hãy lập trình đếm số các phần tử là số **Armstrong** xuất hiện trong dãy A.

Input: Dòng đầu tiên chứa số nguyên dương n thỏa $1 \leq n \leq 2.10^5$.

Dòng thứ hai chứa các số $a_1, a_2, \ldots, a_n$ thỏa $1 \leq a_i \leq 10^9$.

Output: In ra kết quả cần đếm.

Sample Input 1

10

153 1221 100 1634 121 98 12 32 371 125

Sample Output 1

3

Link nộp bài: https://coder.husc.edu.vn/problem/pb0091

Bài toán 5.3. Số đảo ngược là nguyên tố

Hãy viết hai hàm sau:

- Hàm đảo ngược một số nguyên n, ví dụ 1234 sẽ đảo lại thành 4321.

- Hàm kiểm tra xem n có phải là số nguyên tố hay không?

Áp dụng: Với một số nguyên dương cho trước, số viết ngược của nó có phải là một số nguyên tố hay không?

Input: Dòng đầu tiên chứa số nguyên T là số testcase thỏa $1 \leq T \leq 10$.

T dòng tiếp theo mỗi dòng chứa số nguyên dương n thỏa $1 \leq n \leq 10^{12}$.

Output: Ứng với mỗi testcase in ra **Yes** nếu thỏa và **No** nếu ngược lại, mỗi testcase in trên một dòng.

Sample Input 1

2

51

914

Sample Output 1

No

Yes

Link nộp bài: https://coder.husc.edu.vn/problem/pb0092

Bài toán 5.4. Giá trị của đa thức

Hãy viết hàm tính a^n.

Áp dụng: Tính $S = (x^0 - 1) + (x^2) + (x^4) + \ldots + (x^n)$

Input: Dòng duy nhất chứa hai số nguyên x, n thỏa $0 \leq x, n \leq 10$.

Output: In ra kết quả cần tính.

Sample Input 1

5 5

Sample Output 1

650

Link nộp bài: https://coder.husc.edu.vn/problem/pb0093

Bài toán 5.5. Số hoàn hảo

Hãy viết hàm kiểm tra xem một số nguyên dương n có phải là số hoàn hảo hay không? **Số hoàn hảo** là số tự nhiên có tổng các ước không kể nó bằng chính số đó. Ví dụ $6 = 1 + 2 + 3$

Áp dụng: Kiểm tra một số có phải là số hoàn hảo hay không, nếu là số hoàn hảo thì in ra các ước số của nó.

Input: Dòng duy nhất chứa số nguyên n thỏa $0 \leq n \leq 10^6$.

Output: In ra **Yes** ở dòng thứ nhất nếu n là số hoàn hảo, dòng thứ hai in ra danh sách các ước số cách nhau ký tự space. Ngược lại in ra **No**.

Sample Input 1

6

Sample Output 1

Yes

1 2 3

Link nộp bài: https://coder.husc.edu.vn/problem/pb0094

Bài toán 5.6. Tổng dãy xoay chiều

Hãy viết các hàm sau:

- Hàm xoay một dãy $A = \{a_1, a_2, \ldots, a_n\}$ từ **trái sang phải** k đơn vị, ví dụ $A = \{1, 2, 3, 4, 5, 6, 7, 8\}$ với $k = 3$ ta có dãy sau khi xoay là $A = \{6, 7, 8, 1, 2, 3, 4, 5\}$.

- Hàm xoay một dãy $A = \{a_1, a_2, \ldots, a_n\}$ từ **phải sang trái** k đơn vị, ví dụ $A = \{1, 2, 3, 4, 5, 6, 7, 8\}$ với $k = 3$ ta có dãy sau khi xoay là $A = \{4, 5, 6, 7, 8, 1, 2, 3\}$.

Áp dụng: Cho hai dãy là A và B có cùng số phần tử . Tìm dãy C là tổng của A và B khi xoay A từ trái sang phải và xoay B từ phải sang trái k đơn vị.

Input: Dòng đầu tiên gồm hai số nguyên n, k thỏa $1 \leq n \leq 10^5, 0 \leq k < n$. Dòng thứ hai chứa n số nguyên a_i là các phần tử của dãy A thỏa $1 \leq a_i \leq 10000$. Dòng thứ ba chứa n số nguyên b_i là các phần tử của dãy B thỏa $1 \leq b_i \leq 10000$.

Output: In ra dãy C cần tính trên một dòng, các phần tử cách nhau dấu cách.

Sample Input 1

8 3

1 2 3 4 5 6 7 8

1 2 3 4 5 6 7 8

Sample Output 1

10 12 14 8 10 4 6 8

Link nộp bài: https://coder.husc.edu.vn/problem/pb0095

Bài toán 5.7. Xuất hiện nhiều nhất

Hãy viết hàm sau: In ra phần tử xuất hiện nhiều lần nhất trong một dãy n phần tử. Ví dụ dãy $A = \{23, 5, 2, 34, 56, 56, 23, 54, 6, 2\}$ sẽ cho kết quả là 23.

Áp dụng: Cho dãy n phần tử số nguyên A, hãy in ra phần tử xuất hiện nhiều lần nhất.

Chú ý: Nếu có nhiều phần tử xuất hiện nhiều nhất bằng nhau thì in ra phần tử xuất hiện đầu tiên của dãy thỏa điều kiện tìm.

Input: Dòng đầu tiên gồm số nguyên n thỏa $1 \leq n \leq 10^5$. Dòng thứ hai chứa n số nguyên a_i là các phần tử của dãy A thỏa $1 \leq a_i \leq 10^9$.

Output: In ra phần tử cần tìm.

Sample Input 1

10

23 5 2 34 56 56 23 54 6 2

Sample Output 1

23

Link nộp bài: https://coder.husc.edu.vn/problem/pb0096

Bài toán 5.8. Đệ quy bản 1

Hôm nay **Bi** học về lập trình hàm đệ quy (recursive function). Một nhận định mà thầy giáo đưa ra là **Hầu hết các bài toán đều có thể cài đặt bằng đệ quy**. Để làm quen với khái niệm hàm đệ quy thầy yêu cầu **Bi** cài đặt hàm sau:

$$\text{f(n)} = \begin{cases} 0 & \text{nếu } n = 0 \\ f(n-1) + n & \text{nếu } n > 0 \end{cases}$$

Input: Dòng duy nhất số nguyên n thỏa $1 \leq n \leq 10^5$.

Output: In ra kết quả khi gọi hàm cài đặt ở trên.

Sample Input 1

5

Sample Output 1

15

Sample Input 2

10

Sample Output 2

55

Link nộp bài: https://coder.husc.edu.vn/problem/pb0097

Bài toán 5.9. Đệ quy bản 2

Hãy cài đặt hàm sau bằng đệ quy.

$$\text{f(n)} = \begin{cases} a & \text{nếu } n = 0 \\ f(n-1) + bn + c & \text{nếu } n > 0 \end{cases}$$

Input: Dòng duy nhất bốn số nguyên a, b, c, n thỏa $|a|, |b|, |c| \leq 10^3, 0 \leq n \leq 10^3$.

Output: In ra kết quả khi gọi hàm cài đặt ở trên.

Sample Input 1

4 2 -3 10

Sample Output 1

84

Link nộp bài: https://coder.husc.edu.vn/problem/pb0098

Bài toán 5.10. Đệ quy bản 3

Hãy cài đặt hàm sau bằng đệ quy:

$$\mathrm{A(m,n)} = \begin{cases} n+1 & \text{nếu } m = 0 \\ A(m-1,1) & \text{nếu } m > 0, n = 0 \\ A(m-1, A(m,n-1)) & \text{nếu } m > 0, n > 0 \end{cases}$$

Input: Dòng duy nhất hai số nguyên m, n thỏa $0 \leq m \leq 3; 1 \leq n \leq 10^6$.

Chú ý: Nếu $m = 3$ thì $n \leq 24$.

Output: In ra kết quả khi gọi hàm cài đặt ở trên.

Sample Input 1

2 4

Sample Output 1

11

Link nộp bài: https://coder.husc.edu.vn/problem/pb0099

Bài toán 5.11. Đệ quy bản 4

Hãy cài đặt hàm sau bằng đệ quy:

$$\text{f(n)} = \begin{cases} f(f(n+11) & \text{nếu } n \leq 100 \\ n - 10 & \text{nếu } n \geq 101 \end{cases}$$

Input: Dòng duy nhất số nguyên n thỏa $1 \leq n \leq 10^6$.

Output: In ra kết quả khi gọi hàm cài đặt ở trên.

Sample Input 1

91

Sample Output 1

91

Link nộp bài: https://coder.husc.edu.vn/problem/pb0100

Bài toán 5.12. Đệ quy bản 5

Hãy cài đặt hàm sau bằng đệ quy:

$$\mathrm{f(m,n)} = \begin{cases} 1 & \text{nếu } m = 0 \\ 1 & \text{nếu } m = n \\ f(m-1, n-1) + f(m, n-1) & \text{nếu } 0 < m < n \end{cases}$$

Input: Dòng duy nhất hai số nguyên không âm n, m thỏa $0 \leq n, m \leq 20$.

Output: In ra kết quả khi gọi hàm cài đặt ở trên.

Sample Input 1

4 2

Sample Output 1

6

Link nộp bài: https://coder.husc.edu.vn/problem/pb0101

Phần II

Lập trình nâng cao

Tài liệu tham khảo

[1] D. Malik, *C++ Programming: From Problem Analysis To Program Design.* 20 Channel Center Street Boston, MA 02210: Course Technology, 2017.

[2] S. Manuel Rubio, *Introduction to Recursive Programming.* 6000 Broken Sound Parkway NW: CRC Press, 2018.