MỤC LỤC

LỜI NÓI ĐẦU

Lâu nay, việc học lập trình trong một số nhà trường phổ thông nói chung còn chưa được quan tâm đúng mức. Bên cạnh đó việc học lập trình ở các lớp chuyên của các trường chuyên lại được tổ chức học tập theo kiểu đào tạo mũi nhọn nên nhiều khi trong chính lớp chuyên chỉ có một bộ phận theo đuổi học lập trình. Việc học tập như vậy cũng đã tạo ra một thế hệ học sinh rất tài năng và thực sự có nhiều đóng góp cho xã hội. Tuy nhiên, nếu các nhà trường chỉ theo đuổi chinh phục đỉnh cao mà quên đi việc đào tạo nghề sớm và trang bị kỹ năng tin học cho nhiều học sinh khác thì đó sẽ là một chính sách giáo dục sai lầm. Chưa kể thêm là phần nào đó các kỳ thi học sinh giỏi hiện tại không còn nguyên vẹn ý nghĩa trong sáng ban đầu của nó nữa khiến cho học trò học tủ, học lệch mà không nghiên cứu thuật toán một cách trọn vẹn để phục vụ cho nghề nghiệp trong tương lai.

Tác giả là một người tham gia giảng dạy cho chuyên Tin 15 năm, cũng từng vật lộn với từng bài toán dễ đến khó, cũng từng hưởng niềm vui chiến thắng khi học trò đạt các danh hiệu, cũng từng cay đắng khi công sức bỏ ra tan thành mây khói. Hơn hết tác giả hiểu rằng, con đường học vấn chỉ là một phần của cuộc sống, phải học tập làm sao để có thể sống tốt và cống hiến một phần cho xã hội mới là mục đích thực sự của học tập. Vì vậy tác giả cố gắng biên soạn một bộ sách lập trình dành cho học sinh phổ thông để việc học lập trình trở nên dễ dàng hơn chứ không phải là một quá trình đi tìm kiến thức một cách bị động. Từ đó, người học sẽ định hình nghề nghiệp trong tương lai một cách tự nhiên theo năng khiếu của bản thân. Đây là công việc cực kỳ khó khăn nhưng tác giả và các cộng sự vẫn cố gắng quyết tâm theo đuổi mục đích này.

Cuốn sách này là phần mở đầu - quyển Hạ của bộ sách có tên "Tìm hiểu lập trình thi đấu" với 3 quyển: Hạ – Trung – Thượng. Quyển Hạ sẽ là quyển nhập môn lập trình thi đấu với những kiến thức cơ bản nhất về ngôn ngữ lập trình C++ và phương pháp tiếp cận một cách đơn giản dễ hiểu cộng với một web JUDGE có địa chỉ http://laptrinhphothong.vn để học sinh có thể nạp bài và kiểm tra tính đúng đắn của lời giải. Hầu hết các bài tập ở quyển Hạ không có gì mới mẻ cả mà tác giả chỉ chắp bút phân loại các mức độ để học sinh dễ dàng tiếp cận hơn. Có thể trong quá trình sưu tập và sáng tạo bài tập không tránh khỏi sự trùng lặp ý tưởng. Biển học là mênh mông, nếu tài liệu có điểm trùng lặp kính mong bạn đọc bỏ quá vì đây là một tài liệu mang tính chất đại trà phổ biến kiến thức rất khó để viết một cách sáng tạo tuyệt đối. Trong tài liệu này với việc tìm hiểu về thang đánh giá Bloom(2001) tác giả phân loại bài tập theo 5 dạng A – Nhớ, B – Hiểu, C – Vận dụng, D – Phân tích, E – Đánh giá để học sinh phân loại bài tập khi sử dụng tài liệu. Hiện tại tác giả và cộng sự đang sử dụng nền tảng của NTU Coder để xây dựng hệ thống bài tập, trong tương lai không xa sẽ ra mắt web JUDGE riêng. Xin chân thành cảm ơn thầy Trần Minh Văn – admin của NTU Coder đã hỗ trợ nền tảng để tác giả có thể biên soạn được tài liệu.

Hy vọng cuốn sách sẽ giúp ích bạn đọc có được tài liệu tốt để học tập!

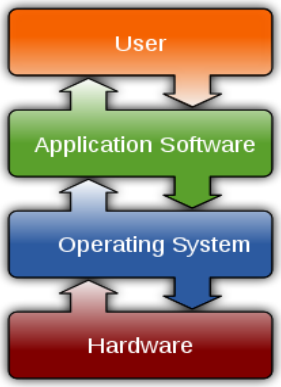
Mọi thông tin góp ý và hợp tác xin gửi về email: nguyenductoandhv@gmail.com.

Trịnh đã viết rằng:

"Cảm ơn đời mỗi ban mai thức dậy, ta có thêm ngày nữa để yêu thương!"

CHƯƠNG 0 – GIỚI THIỆU CODEBLOCKS VÀ C++

1. Sơ lược về lập trình



- Phần mềm máy tính (Software) là tập hợp các chỉ dẫn cho máy tính sao cho máy tính có thể sử dụng các phần cứng (Hardware) hoạt động để thực hiện một nhiệm vụ nào đó.
- Công việc lên kịch bản để máy tính thực hiện một nhiệm vụ nào đó bằng các mã lệnh được gọi là lập trình.
- Người lên kịch bản ở trên được gọi là lập trình viên, kịch bản các mã lệnh được gọi là chương trình.
- Ở cấp độ thấp nhất, chương trình là mã lệnh bằng ngôn ngữ máy – là tập các chỉ thị được CPU trực tiếp thực thi.
- Ở các thế hệ máy tính đầu tiên, để tạo ra chương trình cho máy tính thực hiện, các lập trình viên phải viết chương trình bằng mã máy – đó là một công việc vô cùng khó khăn.
- Hiện tại các lập trình viên đều viết mã lệnh bằng ngôn ngữ lập trình bậc cao – là ngôn ngữ muốn máy tính thực thi cần được chuyển dịch sang ngôn ngữ máy; các ngôn ngữ C++, Python, Java… đều là các ngôn ngữ lập trình bậc cao.
- C++ được tạo ra bởi Stroustrup năm 1979 khi ông nghiên cứu về luận án tiến sỹ của mình; trải qua nhiều phiên bản hiện tại C++ đang là ngôn ngữ hết sức được ưa chuộng trong việc nghiên cứu thuật toán.

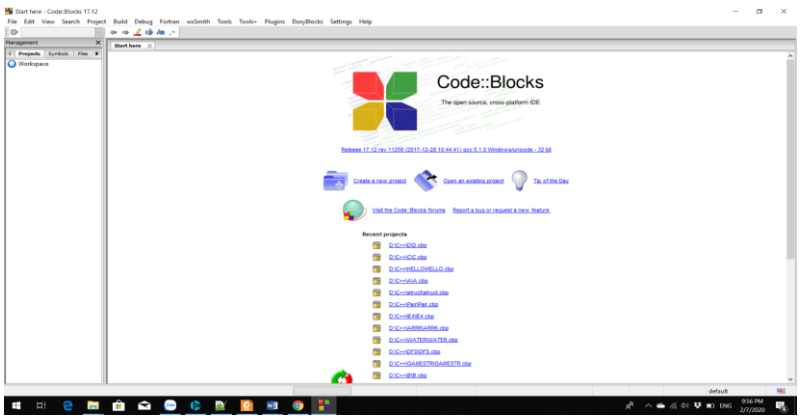
2. Code block và gcc

Để lập trình trên một ngôn ngữ lập trình biên dịch, ta cần có 2 phần mềm cơ bản ngoài hệ điều hành. Một là một trình biên dịch nhằm chuyển đổi chương trình từ ngôn ngữ bậc cao thành ngôn ngữ máy. Mỗi một ngôn ngữ lập trình đều có một trình biên dịch (Compiler) riêng. Hiện tại GNU Compiler Collection (GCC) là một trình biên dịch có thể biên dịch được nhiều ngôn ngữ khác nhau dù mục đích đầu tiên tạo ra chỉ để biên dịch C++. Bên cạnh trình biên dịch, để lập trình thì lập trình viên cần có một công cụ soạn thảo (Text Editor) – đây là phần mềm soạn thảo ra mã lệnh và từ đó chuyển đổi mã lệnh sang mã máy thực thi trực tiếp trên máy tính. Các công cụ soạn thảo ưa dùng hiện nay là Notepad++, Sublime Text, Visual Studio Code …Nếu tích hợp cả hai phần mềm Compiler và Text Editor và một số công cụ hỗ trợ khác thì ta gọi chung là IDE (Integrated Development Environment). Công cụ Codeblocks là một IDE lập trình cho ngôn ngữ C++; ta có thể download CodeBlocks version 20.03 tại trang web codeblocks.org. Chú ý rằng có nhiều phiên bản khác nhau, nếu bạn muốn cài đặt

cả Text Editor lẫn gcc thì cần download bản codeblocks - 20.03 MinGW-setup.exe.

3. Cài đặt Codeblock

- Truy cập website http://codeblocks.org, tìm tới mục download.
- Chọn phiên bản codeblocks-20.03 MinGW-setup.exe để download cho phù hợp với hệ điều hành Windows.
- Với hệ điều hành MAC hay Linux sẽ có các bản cài đặt tương ứng.
- Sau khi download thì tiến hành cài đặt theo tiến trình định sẵn.
- Màn hình làm việc của Codeblock như sau:



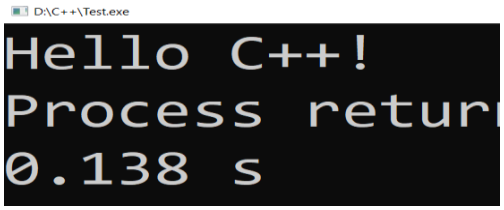
4. Cách chạy một chương trình trên CodeBlocks

- Mở công cụ CodeBlocks;
- Tạo một tệp mới bằng vào menu File\New\Empty File;
- Đặt tên cho file mới là Test.cpp (chú ý rằng ta đang làm việc với C++ nên phải đặt tên file có phần mở rộng là cpp);

- Tiến hành gõ đoạn mã lệnh sau đây vào cửa sổ soạn thảo:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    cout<<"Hello C++!";
    return 0;
}
```

- Nhấn phím F9 để tiến hành biên dịch chương trình sang mã máy và chạy chương trình thu được kết quả sau:



- Như vậy ta đã chạy xong chương trình đầu tiên trên IDE CodeBlocks.

5. Các IDE khác

- Ngoài CodeBlock ta có thể cài đặt một số công cụ khác như Dev – C++ hay Sublime Text, Notepad++, Visual Studio Code…
- Ta cũng có thể sử dụng một số IDE online như https://ideone.com/, https://onecompiler.com/

CHƯƠNG 1 – CÁC KHÁI NIỆM CƠ BẢN CỦA C++

A. KIẾN THỨC GHI NHỚ

1. Khung chương trình của C++

Một chương trình C++ có cấu trúc chung như sau:

```
#include <bits/stdc++.h>//Khai báo thư viện
using namespace std;
//Các loại khai báo khác như chương trình con, biến….
int main()
{
    //Các lệnh chính;
}
```

2. Câu lệnh, khoảng trắng, comment

- Mỗi chương trình bao gồm nhiều câu lệnh hợp thành; câu lệnh là một đơn vị cơ bản của ngôn ngữ lập trình, nó là một tập hợp các chỉ thị được sắp xếp theo trình tự xác định và nhằm mục đích ra lệnh cho CPU tiến hành một thao tác cố định có ý nghĩa;
- Ta có thể phân câu lệnh thành nhiều loại như: câu lệnh khai báo, câu lệnh định nghĩa, câu lệnh tính toán, câu lệnh gán, câu lệnh gọi,…
- Câu lệnh đứng một mình riêng lẻ được gọi là câu lệnh đơn, các câu lệnh được gộp lại với nhau trong dấu {} thì ta gọi là câu lệnh ghép;
- Trong C++ định nghĩa khoảng trắng là các ký tự cách trống và dấu enter;
- Trong chương trình, để giải thích cho các câu lệnh ta thường sử dụng comment được đặt sau hai dấu gạch chéo // , chú ý rằng các comment sẽ bị trình biên dịch bỏ qua khi biên dịch thành mã máy; khi viết chương trình chúng ta cần có comment để giải thích chương trình rõ ràng hơn.

3. Các kiểu dữ liệu chuẩn

Ngôn ngữ C++ cung cấp một số kiểu dữ liệu chuẩn thường dùng sau đây. Chú ý rằng miền giá trị của các kiểu dữ liệu sẽ phụ thuộc vào hệ điều hành và cấu trúc của máy tính. Trong bảng sau miền giá trị xét trên hệ điều hành Windows 32 – bit.

Loại dữ liệu	Tên kiểu	Số ô nhớ	Miền giá trị
Kí tự	char	1 byte	$-128..127$
	unsigned char	1 byte	$0..255$
Số nguyên	int	4 byte	$-2^{15}..2^{15}-1$
	unsigned int	4 byte	$0..2^{16}-1$
	long long	8 byte	$-2^{31}..2^{31}-1$
	unsigned long long	8 byte	$0..2^{32}-1$

Số thực	float	4 byte	Chính xác đến 7 chữ số
	double	8 byte	Chính xác đến 15 chữ số
	long double	16 byte	Chính xác đến 18 chữ số
Logic	bool	1 byte	false – true (false là 0, true là 1)

4. Biến, hằng số, biểu thức và các lệnh vào ra

Biến, hằng số, biểu thức:

- Biến là vùng trống trong bộ nhớ máy tính dành cho một kiểu dữ liệu nào đó và phải có đặt tên;
- Các biến trong bộ nhớ ở các thời điểm khác nhau có thể cất giữ các giá trị khác nhau;
- Trước khi sử dụng một biến nào đó phải khai báo biến đó;
- Cú pháp khai báo biến như sau: `<kiểu dữ liệu> <danh sách biến>;`

```
int a, b;
float x,y,z;
```

- Hằng số là một vùng nhớ trong bộ nhớ máy tính có chứa một giá trị cố định nào đó không thay đổi trong suốt quá trình thực hiện chương trình; khai báo hằng như sau:

```
const int a = 100;
const double pi = 3.14;
```

- Các giá trị biến, hằng tác động với nhau bằng các phép toán tạo nên các biểu thức.

```
S = a+b+c;
double P = sqrt(b*b-4*a*c);
```

Phép toán, lệnh gán:

- Phép toán trên số nguyên:

Phép cộng	A + B
Phép trừ	A - B
Phép nhân	A*B
Phép chia nguyên	A/B
Phép chia dư	A%B

- Phép toán trên số thực:

Phép cộng	A + B
Phép trừ	A - B
Phép nhân	A*B
Phép chia	A/B

- Lệnh gán: lệnh gán là lệnh có dạng `<biến> = <biểu thức>;` có ý nghĩa tính toán giá trị của `biểu thức` rồi đổ dữ liệu đó vào trong `biến`. Ví dụ:

```
int a = 1, b = 2, S;
S = a+b;
```

Lệnh vào ra dữ liệu:

- Với C++, ta có hai câu lệnh vào ra dữ liệu cơ bản là cin và cout.
- Cú pháp của lệnh vào dữ liệu như sau:

  `cin >> (biến 1) >> (biến 2) >> (biến 3);`

  Ví dụ: `cin >> a; cin >> a >> b >> c;`

- Cú pháp của lệnh ra dữ liệu như sau:

  `cout << (biểu thức 1) << (biểu thức 2) << (biểu thức 3);`

  Ví dụ: `cout << a; cin << a + b + c; cout << endl;`

Một số kiểu dữ liệu có thể chuyển đổi cho nhau được như kiểu nguyên và kiểu thực. Để chuyển đổi các kiểu dữ liệu ta có câu lệnh ép kiểu như sau:

- Chuyển số nguyên kiểu int a thành số thực kiểu float a:`(float)a;`
- Cắt phần lẻ của số thực kiểu double b thành kiểu số nguyên long long b: `(long long)b.`
- Bản chất kiểu dữ liệu char cũng là một kiểu số nguyên nên ta có thể thực hiện lệnh ép biến ch kiểu char sang kiểu số số nguyên như sau: int(ch);

B. CÁC VÍ DỤ MẪU

Ví dụ 1.1: Chương trình in ra dòng chào mừng:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    cout<<"Hello C++!";
}
```

Ví dụ 1.2: Chương trình in ra tổng hai số nguyên a, b.

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

Ví dụ 1.3: Chương trình in ra tổng $S = 1 + 2 + \cdots + n$.

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

```
using namespace std;
int main()
{
    long long n;
    cin>>n;
    cout<<n*(n+1)/2;
}
```

Chú ý: Ta dễ dàng chứng minh được $S = n(n+1)/2$ bằng quy nạp toán học.

Ví dụ 1.4: Chương trình tính toán số học:

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    long long a,b; float c;
    cin>>a>>b>>c;
    cout<<a+b<<endl<<a*b<<endl;
    cout<<a/b<<endl<<a%b<<endl;
    cout<<c*(float)a/b;
}
```

Chú ý: Ta có phép chia lấy nguyên giữa hai số nguyên là: a/b, phép chia lấy dư giữa hai số nguyên là $a\%b$, phép chia lấy số thực của số nguyên ta cần ép sang kiểu thực bằng câu lệnh (float)a.

C. BÀI TẬP ÁP DỤNG

Bài 1.1 – (N0101A) Nhập xuất

Yêu cầu: Viết chương trình nhập vào một số nguyên và in ra số nguyên đó

Dữ liệu: Một số nguyên kiểu int.

Kết quả: In ra số nguyên vừa nhập vào.

Ví dụ:

Input	output
10	10

Bài 1.2 – (N0102A) Tính toán 1

Yêu cầu: Viết chương trình in ra tổng hai số nguyên a và b kiểu 64 bit.

Dữ liệu: Một dòng ghi hai số nguyên a, b kiểu 64 bit.

Kết quả: Ghi ra tổng hai số nguyên.

Ví dụ:

input	output
2 3	5

Bài 1.3 – (N0103A) Tính toán 2

Yêu cầu: Viết chương trình nhập vào số nguyên n. In ra số gấp 3 lần số đó.

Dữ liệu: Một dòng ghi một số nguyên n kiểu 64 bit,

Kết quả: Ghi ra số gấp 3 lần n.

Ví dụ:

input	output
3	9

Bài 1.4 – (N0104A) Tính toán 3

Yêu cầu: Viết chương trình in ra tích hai số nguyên a và b kiểu 32 bit.

Dữ liệu: Một dòng ghi hai số nguyên a, b kiểu int.

Kết quả: Ghi ra tích hai số nguyên.

Ví dụ:

input	output
2 3	6

Bài 1.5 – (N0105A) Tính toán 4

Yêu cầu: Tính $S = a * (b + c) + b * (a + c)$.

Dữ liệu: Một dòng ba số nguyên a, b, c ($0 < |a|, |b|, |c| \leq 10^9$).

Kết quả: Một dòng ghi giá trị $S = a * (b + c) + b * (a + c)$.

Ví dụ:

input	output
1 2 3	13

Bài 1.6 – (N0106A) Tính toán 5

Yêu cầu: Viết chương trình nhập vào hai số a, b. In ra $a + b + a * b$.

Dữ liệu: Một dòng ghi 2 số a, b $(0 \leq |a|, |b| < 10^9)$.

Kết quả: Đưa ra kết quả $a + b + a * b$.

Ví dụ:

input	output
1 1	3

Bài 1.7 – (N0107B) Căn n

Yêu cầu: Viết chương trình nhập vào số nguyên dương n $(0 < n \leq 10^{18})$. In ra: $n + \sqrt{n}$. Cho biết hàm lấy phần nguyên căn n là `int(sqrt(n))` hoặc `(long long)(sqrt(n)),` bản chất ở đây là ép kiểu thực sang kiểu nguyên và cắt đi phần lẻ.

Dữ liệu: Một dòng ghi số nguyên n $(0 < n \leq 10^{18})$.

Kết quả: Đưa ra kết quả $n + \sqrt{n}$.

Ví dụ:

input	output
10	13

Bài 1.8 – (N0108B) Tổng 1 đến n

Yêu cầu: Tính tổng $S = 1 + 2 + \cdots n$.

Dữ liệu: Một dòng ghi số nguyên n $(0 < n \leq 10^9)$.

Kết quả: Đưa ra kết quả S.

Ví dụ:

input	output
9	45

Bài 1.9 – (N0109B) Tổng các số lẻ

Yêu cầu: Tính tổng $S = 1 + 3 + \cdots (2n + 1)$.

Dữ liệu: Một dòng ghi số nguyên n $(0 < n \leq 10^9)$.

Kết quả: Đưa ra kết quả S.

Ví dụ:

input	output
3	16

Bài 1.10 – (N0110B) Tổng bình phương

Yêu cầu: Tính tổng $S = 1^2 + 2^2 + \cdots + n^2$.

Dữ liệu: Một dòng ghi số nguyên n $(0 < n \leq 10^6)$.

Kết quả: Đưa ra kết quả S.

Ví dụ:

input	output
3	14

Bài 1.11 – (N0111A) Chia lấy nguyên - dư

Yêu cầu: Viết chương trình nhập vào số nguyên dương n và k . Biết $n = q.k + r$, in ra q và r. Ta có phép chia lấy nguyên trong C++ là n/k và phép chia lấy dư là $n\%k$.

Dữ liệu: Một dòng ghi hai số nguyên n, k $(0 < n, k \leq 10^{18})$.

Kết quả: Đưa ra kết quả q và r trên một dòng.

Ví dụ:

input	output
10 3	3 1

Bài 1.12 – (N0112A) Liền sau – liền trước

Yêu cầu: Viết chương trình nhập vào số nguyên dương n ($0 < n \leq 10^9$). In ra tích của số liền trước và số liền sau của n.

Dữ liệu: Một dòng ghi số nguyên n.

Kết quả: Đưa ra tích của số liền trước và số liền sau của n.

Ví dụ:

input	output
10	99

Bài 1.13 – (N0113A) Phần nguyên – phần lẻ

Yêu cầu: Viết chương trình nhập vào số thực x. In ra phần nguyên và phần lẻ của x. Phần nguyên của x có thể lên tới 10^{18}.

Dữ liệu: Một dòng ghi số thực x.

Kết quả: Đưa ra kết quả là phần nguyên và phần lẻ của x trên một dòng.

Ví dụ:

input	output
1.13	1 0.13

Bài 1.14 – (N0114B) Phép chia

Yêu cầu: Viết chương trình nhập vào 2 số nguyên a, b.

In ra a/b (a chia b lấy nguyên), $a\%b$ (a chia b lấy dư), a/b (a chia b lấy thực, lấy 2 chữ số sau dấu phẩy). Ta chú ý câu lệnh chia lấy số thực như sau:

```
cout << setprecision(2); cout<<fixed; cout<<a/(float)b;
```

Dữ liệu: Một dòng ghi hai số nguyên a, b ($0 < a, b \leq 10^9$).

Kết quả:

- Dòng 1 ghi a/b là kết quả phép chia lấy nguyên,
- Dòng 2 ghi $a\%b$ là kết quả phép chia lấy dư,
- Dòng 3 ghi a/b là kết quả phép chia lấy 2 chữ số sau dấu thập phân.

Ví dụ:

input	output
4 2	2 0 2.00

Bài 1.15 – (N0115C) Tổ hợp

Yêu cầu: Mr X muốn chọn 3 thành viên ban cán sự trong lớp 10A2 có n học sinh. Hỏi số cách chọn?

Dữ liệu: Một dòng ghi số nguyên dương n $(0 < n \leq 10^6)$.

Kết quả: Đưa ra kết quả là số cách chọn.

Ví dụ:

input	output
3	1

Bài 1.16 – (N0116B) Mã ASCII

Yêu cầu: Biết một ký tự trong C++ được khai báo như sau: `char ch;` mỗi một ký tự đều có mã ASCII của nó. Cách in ra ký tự có mã ASCII là n như sau: `cout<<char(n);`

Dữ liệu: Một một số nguyên n $(65 < n \leq 127)$.
Kết quả: Đưa ra ký tự ch có mã ASCII là $n + 1$.
Ví dụ:

input	output
65	B

Bài 1.17 – (N0117C) Chỉnh hợp

Yêu cầu: Mr X muốn chọn 2 thành viên trong lớp 10A2 có n học sinh để cử làm lớp trưởng và lớp phó học tập. Hỏi số cách chọn?

Dữ liệu: Một dòng ghi số nguyên dương n $(0 < n \leq 10^6)$.
Kết quả: Đưa ra kết quả là số cách chọn.
Ví dụ:

input	output
4	12

Bài 1.18 – (N0118C) Giao điểm 1

Yêu cầu: Cho n đường thẳng phân biệt. Hỏi số giao điểm tối đa có thể có của n đường thẳng?

Dữ liệu: Một dòng ghi số nguyên dương n $(0 < n \leq 10^6)$.

Kết quả: Đưa ra kết quả là số giao điểm tối đa.

Ví dụ:

input	output
3	3

Bài 1.19 – (N0119C) Giao điểm 2

Yêu cầu: Cho n đường tròn phân biệt. Hỏi số giao điểm tối đa có thể có của n đường tròn trên?

Dữ liệu: Một dòng ghi số nguyên dương n $(0 < n \leq 10^6)$.

Kết quả: Đưa ra kết quả là số giao điểm tối đa.

Ví dụ:

input	output
4	12

Bài 1.20 – (N0120C) Giao điểm 3

Yêu cầu: Cho m đường tròn và n đường thẳng phân biệt. Hỏi số giao điểm tối đa có thể có của m đường tròn và n đường thẳng trên?

Dữ liệu: Một dòng ghi hai số nguyên dương m, n $(0 < m, n \leq 10^6)$.

Kết quả: Đưa ra kết quả là số giao điểm tối đa.

Ví dụ:

input	output
2 2	11

CHƯƠNG 2 - CẤU TRÚC RẼ NHÁNH

A. KIẾN THỨC GHI NHỚ

1. Cấu trúc rẽ nhánh if

Cú pháp của lệnh if như sau:

```
if (biểu thức điều kiện)
                {
                      <lệnh>;
                }
```

Nếu biểu thức điều kiện đúng thì sẽ thực hiện lệnh, nếu không chương trình sẽ bỏ qua và thực hiện lệnh kế tiếp.

2. Cấu trúc rẽ nhánh if else

Cú pháp của lệnh if như sau:

```
if (biểu thức điều kiện)
                {
                      <lệnh 1>;
                }
else
                {
                      <lệnh 2>;
                }
```

Nếu biểu thức điều kiện đúng thì sẽ thực hiện lệnh thứ nhất, nếu không chương trình sẽ thực hiện lệnh thứ 2.

3. Cấu trúc switch

Cú pháp của lệnh switch như sau:

```
    switch(biểu thức)
    {
        case hằng 1: các lệnh nhóm 1;break;
        case hằng 2: các lệnh nhóm 2;break;
        default: cout<<"OTHER";
    }
```

Nếu biểu thức điều kiện đúng thì sẽ thực hiện lệnh thứ nhất, nếu không chương trình sẽ thực hiện lệnh thứ 2.

4. Các toán tử logic

B. CÁC VÍ DỤ MẪU

Ví dụ 2.1: Đoạn chương trình sau sử dụng cấu trúc `if` để nhập vào một số nguyên in ra 1 nếu số đó là nguyên dương, in ra -1 nếu số đó là nguyên âm và in ra 0 nếu $n = 0$.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    long long n;
    cin>>n;
    if(n>0)  cout<<1;
    if(n==0) cout<<0;
    if(n<0)  cout<<-1;
}
```

Ví dụ 2.2: Chương trình sử dụng cấu trúc `if else` để nhập vào hai số nguyên a, b và in ra số nhỏ hơn.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    long long n;
    cin>>n;
    if(n>0)  cout<<1;
    if(n==0) cout<<0;
    if(n<0)  cout<<-1;
}
```

Ví dụ 2.3: Chương trình sử dụng cú pháp `if else` lồng nhau

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    long long a,b;
    cin>>a>>b;
    if(b==0) cout<<"MATH ERROR";
       else if(a%b==0) cout<<"YES";
               else cout<<"NO";
}
```

Ví dụ 2.4: Chương trình sử dụng cú pháp `switch () case`.

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

```cpp
int main()
{
    long long n;
    cin>>n;
    switch(n)
    {
        case 1: cout<<"ONE"; break;
        case 2: cout<<"TWO"; break;
        default: cout<<"OTHER";
    }
}
```

C. BÀI TẬP ÁP DỤNG

Bài 2.1 – (N0201A) Chẵn lẻ

Yêu cầu: Cho số nguyên n. In ra ODD nếu n lẻ, in ra EVEN nếu n chẵn.

Dữ liệu: Một dòng ghi một số nguyên n $(0 < n \leq 10^9)$.

Kết quả: Ghi ra ODD hoặc EVEN

Ví dụ:

input	output
1	ODD

Bài 2.2 – (N0202A) Chia hết

Yêu cầu: Cho hai số nguyên a, b. In ra YES nếu a chia hết cho b, in ra -1 nếu $b = 0$, in ra NO nếu a không chia hết cho b.

Dữ liệu: Một dòng ghi hai số nguyên a, b là số nguyên kiểu 64 – bit.

Kết quả: Ghi ra YES, -1, NO như yêu cầu đề bài.

Ví dụ:

input	output
1 2	NO

Bài 2.3 – (N0203B) Năm nhuận

Yêu cầu: Năm Nhuận là năm chia hết cho 4 mà không chia hết cho 100, nhưng chia hết cho 400 thì vẫn là năm Nhuận. Nhập một năm, nếu năm đó là năm Nhuận thì in ra YES, nếu không in ra NO.

Dữ liệu: Một dòng ghi một số nguyên là năm cần xét.

Kết quả: Ghi ra YES, NO như yêu cầu đề bài

Ví dụ:

input	output
4	YES

Bài 2.4 – (N0204A) Số lớn hơn 1

Yêu cầu: Nhập vào 2 số phân biệt a, b. In ra số lớn hơn.

Dữ liệu: Một dòng ghi 2 số nguyên a, b là các số nguyên 32 – bit.

Kết quả: In ra số lớn hơn.

Ví dụ:

input	output
2 3	3

Bài 2.5 – (N0205B) Số lớn hơn 2

Yêu cầu: Nhập vào 4 số phân biệt a, b, c, d. In ra số lớn nhất trong 4 số đó.

Dữ liệu: Một dòng ghi 4 số nguyên a, b, c, d là các số nguyên 64 – bit.

Kết quả: In ra giá trị lớn nhất trong 4 số a, b, c, d.

Ví dụ:

input	output
1 2 3 4	4

Bài 2.6 – (N0206A) Số chính phương

Yêu cầu: Nhập vào số nguyên n, in ra YES nếu n là số chính phương, in ra NO nếu n không chính phương. Ta có thể kiểm tra số chính phương bằng cách kiểm tra n có bằng $sqrt(n) * sqrt(n)$ hay không.

Cú pháp so sánh như sau:

```
n==(long long)(sqrt(n))* (long long)(sqrt(n)).
```

Dữ liệu: Một dòng ghi số nguyên n số nguyên 64 – bit.

Kết quả: In ra YES hoặc NO nếu n là chính phương hoặc không tương ứng.

Ví dụ:

input	output
9	YES

Bài 2.7 – (N0207A) Ba cạnh tam giác

Yêu cầu: Nhập vào số nguyên ba số nguyên a, b, c. Kiểm tra xem 3 số đó có lập thành 3 cạnh của tam giác hay không?

Dữ liệu: Một dòng ghi 3 số nguyên a, b, c là số nguyên 64 – bit.

Kết quả: In ra YES hoặc NO nếu 3 số có thể lập thành 3 cạnh của tam giác hay không.

Ví dụ:

input	output
1 1 1	YES

Bài 2.8 – (N0208A) Phương trình bậc hai

Nhập vào số nguyên ba số nguyên a, b, c. Giải phương trình $ax^2 + bx + c = 0$. Đặt $\Delta = b^2 - 4ac$, ta có nếu $\Delta < 0$ thì phương trình vô nghiệm, nếu $\Delta = 0$ thì phương trình có 1 nghiệm, còn khi $\Delta > 0$ thì phương trình có 2 nghiệm phân biệt.

Yêu cầu:

- In ra NOSOL nếu phương trình vô nghiệm,
- In ra ONE nếu phương trình có nghiệm kép,
- In ra TWO nếu có 2 nghiệm phân biệt.

Dữ liệu: Một dòng ghi 3 số nguyên a, b, c là số nguyên 64 - bit (a khác 0)

Kết quả: In ra NOSOL, ONE, TWO theo yêu cầu đề bài.

Ví dụ:

input	output
1 1 1	NOSOL

Bài 2.9 – (N0209A) Số nhỏ nhất 1

Yêu cầu: Cho ba số nguyên a, b, c ($|a|, |b|, |c| \leq 10^9$), tìm giá trị nhỏ nhất của ba số a, b, c.

Dữ liệu: Một dòng ghi ba số nguyên a, b, c.

Kết quả: Đưa ra giá trị nhỏ nhất trong ba số a, b, c.

Ví dụ:

input	output
1 3 4	1

Bài 2.10 – (N0210A) Số nhỏ nhất 2

Yêu cầu: Cho ba số nguyên a, b, c ($|a|, |b|, |c| \leq 10^9$), tìm giá trị nhỏ nhất của bốn số a, b, c, d.

Dữ liệu: Một dòng ghi bốn số nguyên a, b, c, d.

Kết quả: Đưa ra giá trị nhỏ nhất trong bốn số a, b, c, d.

Ví dụ:

input	output
1 3 5 7	1

Bài 2.11 – (N0211B) Ba điểm thẳng hàng

Trong hệ trục tọa độ Oxy, ba điểm A, B, C thẳng hàng nếu $\overrightarrow{AB} = k\overrightarrow{AC}$ hay tọa độ ba điểm tương ứng tỷ lệ với nhau như sau:

$$(x_B - x_A)(y_C - y_A) = (x_C - x_A)(y_B - y_A).$$

Yêu cầu: Cho tọa độ 3 điểm A, B, C. Kiểm tra xem 3 điểm A, B, C có thẳng hàng không?

Dữ liệu: Một dòng ghi 6 số nguyên xa, ya, xb, yb, xc, yc có kiểu dữ liệu int là tọa độ của A, B, C.

Kết quả: Đưa ra YES nếu A, B, C thẳng hàng, NO nếu A, B, C không thẳng hàng.

Ví dụ:

input	output
1 1 2 2 3 3	YES

Bài 2.12 – (N0212B) Chia hết cho 3

Yêu cầu: Cho hai số a, b. Kiểm tra xem 2 chữ số cuối cùng của tích $a \times b$ có chia hết cho 3 hay không?

Dữ liệu: Một dòng ghi hai số nguyên a, b, c ($|a|, |b|, |c| \leq 10^9$).

Kết quả: Ghi YES nếu hai chữ số cuối cùng của $a * b$ chia hết cho 3, ghi NO nếu ngược lại.

Ví dụ:

input	output
10 10	YES

Bài 2.13 – (N0213B) Số nhỏ nhì

Yêu cầu: Cho 5 số nguyên a, b, c, d, e kiểu int đôi một khác nhau. In ra số nhỏ thứ nhì.

Dữ liệu: Một dòng gồm 5 số nguyên a, b, c, d, e.

Kết quả: In ra số nhỏ thứ nhì.

Ví dụ:

input	output
1 2 3 4 5	2

Bài 2.14 – (N0214B) Quy tắc chia

Yêu cầu: Cho 3 số nguyên a, b, c. In ra dấu / nếu $a/b = c$ hoặc $b/c = a$ hoặc $c/a = b$ và in ra NOSOL nếu không thỏa mãn.

Dữ liệu: Một dòng gồm 3 số nguyên a, b, c ($|a|, |b|, |c| \leq 10^9$).

Kết quả: Ghi ra / nếu $a/b = c$ hoặc $b/c = a$ hoặc $c/a = b$ in ra NOSOL nếu không thỏa mãn.

Ví dụ:

input	output
1 2 2	/

Bài 2.15 – (N0215A) Bằng nhau

Yêu cầu: Cho 5 số kiểu int. In ra YES nếu có ít nhất 4 số bằng nhau, in ra NO nếu không thỏa mãn.

Dữ liệu: Một dòng gồm 5 số nguyên a, b, c, d, e $(0 < |a|, |b|, |c|, |d|, |e| \leq 10^9)$.

Kết quả: Ghi ra YES nếu có 4 số bằng nhau, ghi NO nếu ngược lại.

Ví dụ:

input	output
1 1 1 1 2	YES

Bài 2.16 – (N0216D) Ngày sinh

Yêu cầu: Cho ngày sinh của một người. In ra thứ của ngày sinh người đó.

Dữ liệu: Một dòng gồm 3 số nguyên d, m, y $(0 < d, m, y \leq 10^9)$ là ngày, tháng, năm sinh.

Kết quả: Ghi ra thứ bằng tiếng Anh tương ứng (Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday).

Ví dụ:

input	output
3 9 2019	Tuesday

Bài 2.17 – (N0217C) Nhiều tuổi hơn

Yêu cầu: Cho ngày tháng năm sinh của hai người. In ra 1 nếu người thứ nhất nhiều tuổi hơn người thứ hai, in ra -1 nếu người thứ hai nhiều tuổi hơn người thứ nhất, in ra 0 nếu cả hai bằng tuổi nhau.

Dữ liệu:

- Dòng 1 ghi 3 số nguyên kiểu int $d1, m1, y1$ là ngày tháng năm sinh của người thứ nhất.
- Dòng 2 ghi 3 số nguyên kiểu int $d2, m2, y2$ là ngày tháng năm sinh của người thứ nhất.

Kết quả: in ra kết quả tương ứng $-1, 0, 1$.

Ví dụ:

input	output
1 2 2004 1 3 2004	1

CHƯƠNG 3 - CẤU TRÚC VÒNG LẶP

A. KIẾN THỨC GHI NHỚ

1. Cấu trúc vòng lặp for

Cú pháp của lệnh `for` như sau:

```
for (<biểu thức 1>;<biểu thức 2>;<biểu thức 3>)
{
            <lệnh>;
}
```

Trong đó `<biểu thức 1>` là biểu thức khởi tạo biến lặp, `<biểu thức 2>` là biểu thức dừng vòng lặp, `<biểu thức 3>` là điều khiển lặp.

Ví dụ đơn giản tính tổng $S = 1 + 2 + \cdots + 10$.

```
for (int i=1;i<=10;i=i+1) S = S+i;
```

2. Cấu trúc vòng lặp while

Cú pháp của lệnh `while` như sau:

```
while (<biểu thức điều kiện>)
{
  <lệnh 1>;
  <lệnh 2>;
     …
  <lệnh n>;
}
```

Nếu biểu thức điều kiện đúng thì sẽ thực hiện tất cả các lệnh. Vòng lặp sẽ dừng khi biểu thức điều kiện có kết quả sai.

3. Cấu trúc vòng lặp do while

Cú pháp của lệnh `do while` như sau:

```
do
{
  <lệnh 1>;
  <lệnh 2>;
     …
  <lệnh n>;
}
while(biểu thức điều kiện);
```

Vòng lặp do while sẽ lần lượt thực hiện các câu lệnh từ `<lệnh_1>` đến `<lệnh_n>` và sau đó kiểm tra xem biểu thức điều kiện có đúng không để tiếp tục lặp hoặc thoát ra.

B. CÁC VÍ DỤ MẪU

Ví dụ 3.1: Chương trình sau sử dụng vòng lặp for tính $n!$.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    long long n,S = 1;
    cin>>n;
    for(int i=1;i<=n;i++)
        S*=i;
    cout<<S;
}
```

Ví dụ 3.2: Chương trình sử dụng vòng lặp for lồng nhau in ra bảng cửu chương.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    for(int i=1;i<=9;i++)
    {
        cout<<"Bang nhan "<<i<<endl;
        for(int j=1;j<=9;j++)
            cout<<i<<" x "<<j<<" = "<<i*j<<endl;
    }
}
```

Ví dụ 3.3: Chương trình sau sử dụng vòng lặp while để in ra số nguyên dương nhỏ nhất sao tổng $S = 1 + \frac{1}{2} + \frac{1}{3} + \cdots + \frac{1}{n} > A$ với A là một số cho trước.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    float A, n = 0, S = 0;
    cin>>A;
    while(S<A)
    {
        ++n;
        S=S+1/n;
    }
    cout<<n;}
```

Ví dụ 3.4: Chương trình sau sử dụng vòng lặp do while để đọc dữ liệu từ bàn phím cho đến khi phím Y được nhấn.

```
#include <bits/stdc++.h>
using namespace std;
int main()
{
    char ch;
    do { cin>>ch;}
    while (ch!='Y');
}
```

C. BÀI TẬP ÁP DỤNG

Bài 3.1 – (N0301A) Dấu sao

Yêu cầu: Viết chương trình nhập vào số nguyên n $(0 < n \leq 100)$. In ra n dấu * trên cùng một dòng.

Dữ liệu: Một dòng ghi số nguyên n.

Kết quả: Đưa ra n dấu * viết liền nhau.

Ví dụ:

input	output
4	****

Bài 3.1 – (N0302A) Số tự nhiên

Yêu cầu: Nhập vào số nguyên dương n $(0 < n \leq 10^5)$. In ra các số nguyên dương từ 1 đến n.

Dữ liệu: Một dòng ghi số n duy nhất.

Kết quả: In ra các số nguyên dương từ 1 đến n mỗi số cách nhau 1 dấu cách trống, kết thúc bằng dấu !

Ví dụ:

input	output
5	1 2 3 4 5 !

Bài 3.2 – (N0303A) Dấu thăng

Yêu cầu: Nhập vào số nguyên dương $n(0 < n \leq 10^5)$. In ra 3 hàng dấu #.

Dữ liệu: Một dòng ghi số n duy nhất.

Kết quả: Dòng 1 ghi n dấu #, dòng 2 ghi $n-1$ dấu #, dòng 3 ghi $n-2$ dấu #.

Ví dụ:

input	output

4	#### ### ##

Bài 3.3 – (N0304A) **Dấu đô la**

Yêu cầu: Nhập vào số nguyên dương n $(2 < n \leq 10^3)$. In ra một hình vuông cạnh n gồm các dấu $.

Dữ liệu: Một dòng ghi số n duy nhất.

Kết quả: In ra một hình vuông gồm n^2 dấu $.

Ví dụ:

input	output
4	$$$$ $$$$ $$$$ $$$$

Bài 3.4 – (N0305B) **Số không**

Yêu cầu: Nhập vào số nguyên dương n $(2 < n \leq 10^3)$. In ra một tam giác vuông cân cạnh gồm n các số 0 với dòng cuối cùng là cạnh đáy gồm n số 0.

Dữ liệu: Một dòng ghi số n duy nhất.

Kết quả: Dòng 1 ghi 1 số 0, dòng thứ 2 ghi 2 số 0,…, dòng thứ i ghi i số 0.

Ví dụ:

input	output
4	0 00 000 0000

Bài 3.5 – (N0306B) **Giai thừa**

Yêu cầu: Nhập vào số nguyên dương n $(2 < n \leq 10^3)$. In ra $n!$ chia lấy dư cho $10^9 + 7$.

Dữ liệu: Một dòng ghi số n duy nhất.

Kết quả: Ghi ra kết quả $(n!)\%(10^9 + 7)$.

Ví dụ:

input	output
4	24

Bài 3.6 – (N0307C) **Số dãy nhị phân**

Yêu cầu: Nhập vào số nguyên dương n $(2 < n \leq 15)$.. In ra số dãy nhị phân độ dài n. Chẳng hạn với $n = 2$ ta có các dãy nhị phân là 00, 01, 10, 11. Với $n = 3$ ta có các dãy nhị phân 000, 001, 010, 011, 100, 101, 110, 111.

Dữ liệu: Một dòng ghi số n duy nhất.

Kết quả: Ghi ra số lượng dãy nhị phân độ dài n.

Ví dụ:

input	output
4	16

Bài 3.7 – (N0308B) Dãy số 1

Cho dãy số: $\begin{cases} u_1 = 1, u_2 = 1, \\ u_n = 2u_{n-1} - u_{n-2}, \forall n \geq 3 \end{cases}$

Yêu cầu: Nhập vào số nguyên dương n $(0 < n \leq 10^6)$. In ra số thứ n của dãy.

Dữ liệu: Một dòng ghi số n duy nhất.

Kết quả: Ghi ra số thứ n của dãy số.

Ví dụ:

input	output
4	1

Bài 3.8 – (N0309C) Dãy số 2

Cho dãy số: $\begin{cases} u_1 = 1, u_2 = 1, \\ u_n = 3u_{n-1} - u_{n-2}, \forall n \geq 3 \end{cases}$

Yêu cầu: Nhập vào số nguyên dương n $(2 < n \leq 10^6)$. In ra số thứ n của dãy. Kết quả có thể rất lớn nên ta chia lấy dư cho $10^9 + 7$.

Dữ liệu: Một dòng ghi số n duy nhất.

Kết quả: Ghi ra số thứ n của dãy số chia lấy dư cho $10^9 + 7$.

Ví dụ:

input	output
3	2

Bài 3.9 – (N0310C) Tổng giai thừa

Yêu cầu: Tính tổng $S = 1! + 2! + \cdots + n!$. Kết quả có thể rất lớn nên sẽ chia lấy dư cho $10^9 + 7$.

Dữ liệu: Một dòng ghi số n $(0 < n \leq 10^6)$ duy nhất.

Kết quả: Ghi ra S tương ứng chia lấy dư cho $10^9 + 7$.

Ví dụ:

input	output
2	3

Bài 3.10 – (N0311B) Bảng ký tự

Yêu cầu: Nhập vào hai số nguyên dương m, n $(0 < m, n \leq 10^3)$ và một ký tự ch. In ra bảng m hàng, n cột ký tự vừa nhập.

Dữ liệu:

- Dòng 1 ghi hai số nguyên m, n;
- Dòng 2 ghi ký tự ch.

Kết quả: Ghi ra bảng m hàng, n cột ký tự ch.

Ví dụ:

input	output
2 3 a	aaa aaa

Bài 3.12 – (N0312A) Tổng mũ bốn

Yêu cầu: Viết chương trình in ra tổng $S = 1^4 + 2^4 + \cdots + n^4$.

Dữ liệu: Một dòng ghi số nguyên n $(0 < n \leq 100)$.

Kết quả: Đưa ra tổng S.

Ví dụ:

input	output
2	17

Bài 3.13 – (N0313B) Tổng mũ ba

Yêu cầu: Viết chương trình in ra tổng $S = 1^3 + 2^3 + \cdots + n^3$.

Dữ liệu: Một dòng ghi số nguyên n $(0 < n \leq 10^6)$.

Kết quả: Đưa ra tổng S chia lấy dư cho $10^9 + 7$.

Ví dụ:

input	output
3	36

Bài 3.14 – (N0314B) Tổng đầu

Yêu cầu: Tìm số nguyên k sao cho tổng từ 1 đến k bằng n cho trước.

Dữ liệu: Một dòng ghi số nguyên n $(0 < n \leq 10^6)$.

Kết quả: In ra YES nếu tồn tại k, in ra NO nếu không tồn tại k.

Ví dụ:

input	output
10	YES

Bài 3.15 – (N0315B) Tổng không biết trước số phần tử

Yêu cầu: Tính tổng của một dãy số nguyên chưa biết trước số phần tử.

Dữ liệu: Một dòng ghi không quá 10^5 phần tử là các số nguyên 32 bit cách nhau ít nhất một dấu cách trống.

Kết quả: Ghi ra kết quả là tổng của các số nguyên trên.

Ví dụ:

input	output
1 2 3 4 5 6	21

Bài 3.16 – (N0316C) Số Fibonacci

Yêu cầu: Dãy số Fibonacci là một dãy số có công thức như sau:

$$\begin{cases} u_1 = u_2 = 1, \\ u_n = u_{n-1} + u_{n-2}, \forall n \geq 3 \end{cases}.$$

Hãy tính phần tử thứ n của dãy Fibonacci, kết quả có thể sẽ vượt giá trị của kiểu nguyên 64 bit nên ta chia lấy dư cho $10^9 + 7$ khi in ra.

Dữ liệu: Một dòng ghi một số nguyên dương n ($0 < n \leq 10^6$).

Kết quả: Ghi ra kết quả là số Fibonacci thứ n chia lấy dư cho $10^9 + 7$.

Ví dụ:

input	output
4	3

Bài 3.17 – (N0317C) Tổng không vượt quá A

Yêu cầu: Cho $S = 1 + 1/2 + 1/3 + \ldots + 1/n$. Với số thực A hãy tìm số nguyên dương n nhỏ nhất sao cho $S > A$.

Dữ liệu: Một dòng ghi số A duy nhất.

Kết quả: Ghi ra kết quả số nguyên dương n nhỏ nhất sao cho $S > A$.

Dữ liệu đảm bảo số bước lặp không quá 10^6.

Ví dụ:

input	output
2	4

CHƯƠNG 4 – KIỂU DỮ LIỆU MẢNG MỘT CHIỀU

A. KIẾN THỨC GHI NHỚ

1. Khai báo mảng

Cú pháp khai báo mảng như sau:

```
<tên kiểu dữ liệu> <tên mảng>[số_pt];
```

Trong đó `<tên kiểu dữ liệu>` là kiểu nguyên int, long long hoặc kiểu thực float, double hay kiểu ký tự…Ví dụ ta khai báo: `int a[100];` sẽ cho ta 100 phần tử từ `a[0]` đến `a[99]`.

2. Cách sử dụng

Sau khi khai báo mảng ta truy cập phần tử thông qua chỉ số của phần tử như sau: `<tên_mảng> [chỉ_số]`, ví dụ muốn truy cập phần tử thứ 5 của mảng `a` ta viết `a[5]`.

Để đọc dữ liệu cho mảng ta kết hợp vòng lặp để đọc cho từng phần tử như ví dụ sau:

```
for(int i=1;i<=10;i++) cin>>a[i];
```

Để in dữ liệu mảng ta kết hợp vòng lặp để in cho từng phần tử như ví dụ sau:

```
for(int i=1;i<=10;i++) cout<<a[i];
```

Muốn tác động tới từng phần tử của mảng ta truy cập qua chỉ số như ví dụ sau:

```
for(int i=1;i<=10;i++) a[i]=a[i]+3;
```

B. CÁC VÍ DỤ MẪU

Ví dụ 4.1: Chương trình nhập dữ liệu mảng một chiều 100 phần tử, in ra tổng các phần tử.

```
#include <bits/stdc++.h>
using namespace std;
int a[105],n,S = 0;
int main()
{
      long long n;
      cin>>n;
      for(int i=1;i<=n;i++)
                {
                    cin>>a[i];
                    S = S + a[i];
                 }
       cout<<S;
}
```

Ví dụ 4.2: Chương trình nhập dữ liệu mảng một chiều 100 phần tử, in ra phần tử nhỏ nhất.

```
#include <bits/stdc++.h>
using namespace std;
int a[105],n,NN = 1e9;
int main()
{
      long long n;
      cin>>n;
      for(int i=1;i<=n;i++)
                {
                    cin>>a[i];
                    if(NN>a[i])NN=a[i];
                 }
       cout<<NN;
}
```

Ví dụ 4.3: Chương trình nhập vào mảng một chiều 100 phần tử, đếm số số chẵn.

```
#include <bits/stdc++.h>
using namespace std;
int a[105],n,dem = 0;
int main()
{
      long long n;
      cin>>n;
      for(int i=1;i<=n;i++)
                {
                    cin>>a[i];
                    if(a[i]%2==0) dem++;
                 }
       cout<<dem;
}
```

C. BÀI TẬP ÁP DỤNG

Bài 4.1 – (N0401A) Giá trị nhỏ nhất

Yêu cầu: Viết chương trình nhập vào n $(0 < n \leq 100)$ số nguyên kiểu 64 bit. In ra giá trị nhỏ nhất trong n số nguyên đó.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n số nguyên dương cách nhau một dấu cách trống.

Kết quả: Đưa ra giá trị nhỏ nhất của n số nguyên.

Ví dụ:

input	output

4 2 3 4 6	2

Bài 4.2 – (N0402A) Tổng mảng

Yêu cầu: Viết chương trình nhập vào n $(0 < n \leq 100)$ số nguyên. In ra tổng của n số nguyên đó.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n số nguyên dương cách nhau một dấu cách trống.

Kết quả: Đưa ra tổng của n số nguyên.

Ví dụ:

input	output
4 2 3 4 6	15

Bài 4.3 – (N0403A) Tổng lẻ

Yêu cầu: Viết chương trình nhập vào một dãy gồm n $(0 < n \leq 100)$ số nguyên. In ra tổng các số lẻ trong dãy số nhập vào.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n số nguyên dương cách nhau một dấu cách trống.

Kết quả: Đưa ra tổng các số lẻ của n số nguyên.

Ví dụ:

input	output
4 2 3 4 5	8

Bài 4.4 – (N0404B) Giá trị lớn nhất

Yêu cầu: Viết chương trình nhập vào một dãy gồm n $(0 < n \leq 100)$ số nguyên. In ra giá trị lớn nhất trong n số nguyên đó.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n số nguyên kiểu 64 bit cách nhau một dấu cách trống.

Kết quả: Đưa ra giá trị lớn nhất của n số nguyên đó.

Ví dụ:

input	output
4 2 3 4 5	5

Bài 4.5 – (N0405B) Giá trị lẻ nhỏ nhất

Yêu cầu: Viết chương trình nhập vào một dãy gồm n $(0 < n \leq 100)$ số nguyên. In ra giá trị lẻ nhỏ nhất trong dãy số nhập vào.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n số nguyên dương 64 bit cách nhau một dấu cách trống.

Kết quả: Đưa ra giá trị lẻ nhỏ nhất của n số nguyên.

Ví dụ:

input	output
4 2 3 4 7	3

Bài 4.6 – (N0406A) Giá trị chia hết cho 3

Yêu cầu: Viết chương trình nhập vào một dãy gồm n $(0 < n \leq 100)$ số nguyên. In ra số lượng số chia hết cho 3 trong n số nguyên đó.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n số nguyên dương 64 bit cách nhau một dấu cách trống.

Kết quả: Đưa ra số các số chia hết cho 3.

Ví dụ:

input	output
4 2 3 4 7	1

Bài 4.7 – (N0407A) Tổng trị tuyệt đối

Yêu cầu: Viết chương trình nhập vào n $(0 < n \leq 100)$ số nguyên. In ra tổng các giá trị tuyệt đối trong dãy số.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n số nguyên dương 64 bit cách nhau một dấu cách trống.

Kết quả: Đưa ra tổng các giá trị tuyệt đối của n số nguyên trong dãy.

Ví dụ:

input	output
4 -2 3 -4 7	16

Bài 4.8 – (N0408A) Các số không nhỏ hơn X

Yêu cầu: Viết chương trình nhập vào một dãy n $(0 < n \leq 100)$ số nguyên và số nguyên X. In ra số lượng các số không nhỏ hơn X trong dãy số đó.

Dữ liệu:

- Dòng đầu tiên ghi n và là số nguyên 64 bit,
- Dòng 2 ghi n số nguyên dương 64 bit cách nhau một dấu cách trống.

Kết quả: Đưa ra số các số không nhỏ hơn X trong dãy.

Ví dụ:

input	output
4 1 -2 3 -4 7	2

Bài 4.9 – (N0409A) Số thuộc đoạn

Yêu cầu: Viết chương trình nhập vào n $(0 < n \leq 100)$ số nguyên. In ra số lượng các số thuộc đoạn $[500; 3000]$ trong dãy số.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n số nguyên dương cách nhau một dấu cách trống.

Kết quả: Đưa ra số lượng các số thuộc đoạn $[500; 3000]$ trong dãy số.

Ví dụ:

input	Output
5 501 1 4 600 200	3

Bài 4.10 – (N0410B) Số chính phương

Yêu cầu: Một số a được gọi là chính phương nếu tồn tại số nguyên b thỏa mãn $b^2 = a$. Ta có thể sử dụng một số lệnh để kiểm tra một số có chính phương hay không như sau:

```
c==(int)(sqrt(a)); hoặc c==(long long)(sqrt(n))
if(c*c==a) <a là chính phương>;
else <a không là chính phương>;
```

Viết chương trình nhập vào n $(0 < n \leq 100)$ số nguyên. In ra số lượng các số chính phương trong dãy.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên dương n,
- Dòng 2 ghi n số nguyên dương cách nhau một dấu cách trống.

Kết quả: Đưa ra số lượng các số chính phương trong dãy.

Ví dụ:

input	output
5 1 3 4 9 20	3

Bài 4.11 – (N0411B) Đếm nghịch thế

Yêu cầu: Viết chương trình nhập vào n $(0 < n \leq 100)$ số nguyên. In ra số lượng các cặp $a[i]$ và $a[j]$ thỏa mãn $i < j$ và $a[i] > a[j]$ trong dãy số.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n số nguyên dương cách nhau một dấu cách trống.

Kết quả: Đưa ra số lượng các cặp $a[i]$ và $a[j]$ thỏa mãn $i < j$ và a$[i] > a[j]$.

Ví dụ:

input	output
5 1 20 4 9 5	4

Bài 4.12 – (N0412B) Sắp xếp

Yêu cầu: Viết chương trình nhập vào n $(0 < n \leq 100)$ số nguyên. Sắp xếp theo thứ tự tăng dần các số nguyên đó

Dữ liệu:

- Dòng đầu tiên ghi số nguyên dương n,
- Dòng 2 ghi n số nguyên dương 64 bit cách nhau một dấu cách trống.

Kết quả: Một dòng ghi n số nguyên theo thứ tự tăng dần cách nhau một dấu cách trống. Yêu cầu không được sử dụng hàm sort mà hãy sắp xếp bằng thuật toán "nổi bọt".

Ví dụ:

input	output
5 1 20 4 9 5	1 4 5 9 20

Bài 4.13 – (N0413B) Tổng bình phương

Yêu cầu: Viết chương trình nhập vào một dãy gồm n $(0 < n \leq 10^3)$ số nguyên. In ra giá trị lớn nhất của tổng bình phương $a_i^2 + a_j^2, 0 < i < j \leq n$ trong dãy số.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n số nguyên 64 bit cách nhau một dấu cách trống.

Kết quả: Đưa ra số lượng giá trị lớn nhất của tổng bình phương $a_i^2 + a_j^2, 1 \leq i < j \leq n$ trong dãy số.

Ví dụ:

input	output
5 1 2 3 4 5	41

Bài 4.14 – (N0414C) Quicksort

Yêu cầu: Viết chương trình nhập vào n ($0 < n \leq 10^5$) số nguyên. Sắp xếp theo thứ tự tăng dần các số nguyên đó. Sử dụng hàm `sort(a+1,a+n+1)` với độ phức tạp thuật toán $O(nlogn)$.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên dương n,
- Dòng 2 ghi n số nguyên dương 64 bit cách nhau một dấu cách trống.

Kết quả: Một dòng ghi n số nguyên theo thứ tự tăng dần.

Ví dụ:

input	output
5 6 2 3 4 5	2 3 4 5 6

Bài 4.15 – (N0415B) Số lần xuất hiện nhiều nhất

Yêu cầu: Viết chương trình nhập vào n ($0 < n \leq 100$) số nguyên a_i ($0 \leq a_i \leq 100$). In ra số lần xuất hiện của số xuất hiện nhiều nhất trong dãy số trên.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n số nguyên dương 64 bit cách nhau một dấu cách trống.

Kết quả: Đưa ra số lần xuất hiện của số xuất hiện nhiều nhất trong dãy số trên.

Ví dụ:

input	output
5 6 2 3 4 6	2

Bài 4.16 – (N0416A) Chia hết cho 3

Yêu cầu: Viết chương trình nhập vào n ($0 < n \leq 10^5$) số nguyên. Xóa các phần tử chia hết cho 3 của dãy và in ra dãy sau khi xóa.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên dương n,
- Dòng 2 ghi n số nguyên dương cách nhau một dấu cách trống.

Kết quả: Một dòng ghi dãy sau khi xóa đi các phần tử chia hết cho 3.

Ví dụ:

input	output
5 6 2 3 4 6	2 4

Bài 4.17 – (N0417B) Ký tự xuất hiện nhiều nhất

Yêu cầu: Viết chương trình nhập vào n ($0 < n \leq 100$) ký tự latinh thường ('a' ->'z'). In ra ký tự xuất hiện nhiều nhất trong n ký tự đã cho.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n ký tự cách nhau một dấu cách trống.

Kết quả: Đưa ra ký tự xuất hiện nhiều nhất trong dãy ký tự trên, nếu có 2 ký tự có cùng số lần xuất hiện thì in ra ký tự có thứ tự từ điển nhỏ hơn.

Ví dụ:

input	output
5 a a b b d	a

Bài 4.18 – (N0418B) Chia hết 3 và 5

Yêu cầu: Viết chương trình nhập vào n ($0 < n \leq 10^5$) số nguyên. Xóa các phần tử chia hết cho 3 của dãy và in ra dãy sau khi xóa. Sau đó lại xóa các phần tử chia hết cho 5 trong dãy còn lại và in ra dãy

Dữ liệu:

- Dòng đầu tiên ghi số nguyên dương n,
- Dòng 2 ghi n số nguyên dương 64 bit cách nhau một dấu cách trống.

Kết quả:

- Dòng đầu tiên ghi dãy sau khi xóa đi các phần tử chia hết cho 3,
- Dòng thứ hai ghi dãy sau khi xóa hết các phần tử chia hết cho cả 3 và 5.

Ví dụ:

input	output
7 1 2 3 4 5 6 10	1 2 4 5 10 1 2 4

Bài 4.19 – (N0419B) Giá trị nhỏ nhất đến k

Yêu cầu: Viết chương trình nhập vào n ($0 < n \leq 100$) số nguyên a_i. Tìm giá trị nhỏ nhất của dãy số từ phần tử thứ 1 đến phần tử thứ k nào đó.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n,
- Dòng 2 ghi n ký tự cách nhau một dấu cách trống,
- Dòng 3 ghi số câu hỏi t,
- t dòng sau mỗi dòng ghi một số nguyên k.
- Các số đều là số nguyên 32 bit.

Kết quả: Đưa ra t dòng, mỗi dòng ghi một số nguyên ***mink*** là giá trị nhỏ nhất trong các số a_1 đến a_k tương ứng.

Ví dụ:

input	output
5 5 2 3 4 1 3 2 4 5	2 2 1

Bài 4.20 – (N0420C) Trộn mảng

Yêu cầu: Viết chương trình nhập vào 2 mảng A và B mỗi mảng có n ($0 < n \leq 10^5$) số nguyên. Trộn hai mảng lại thành một mảng và sắp xếp theo thứ tự giảm dần và in ra.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên dương n,
- Dòng 2 ghi n số nguyên dương cách nhau một dấu cách trống biểu diễn mảng A,
- Dòng 3 ghi n số nguyên dương cách nhau một dấu cách trống biểu diễn mảng B.

Kết quả: Một dòng ghi mảng C là trộn hai mảng trên lại và sắp xếp theo thứ tự giảm dần.

Ví dụ:

input	output
5 1 3 2 5 6 2 3 4 5 6	6 6 5 5 4 3 3 2 2 1

Bài 4.21 – (N0421D) Phần tử trung vị

Yêu cầu: Viết chương trình nhập vào một mảng, in ra số trung vị của mảng. Số trung vị là số có giá trị trung bình trong mảng. Nếu mảng có $2n + 1$ phần tử thì sau khi sắp xếp phần tử trung vị là phần tử thứ $n + 1$. Nếu mảng có $2n$ phần tử thì số trung vị là trung bình cộng của hai phần tử ở giữa sau khi sắp xếp mảng.

Dữ liệu:

- Dòng 1 ghi số nguyên dương n ($n \leq 10^5$)
- Dòng 2 ghi n số nguyên cách nhau một dấu cách trống

Kết quả: Một số duy nhất là phần tử trung vị của mảng.

Ví dụ:

input	output
6 1 3 5 2 4 6	3.5

CHƯƠNG 5 – KIỂU DỮ LIỆU MẢNG HAI CHIỀU

A. KIẾN THỨC GHI NHỚ

1. Khái niệm ma trận

Ma trận là một khái niệm toán học cơ bản. Theo định nghĩa của đại số tuyến tính, ma trận là một bảng số gồm m hàng và n cột. Từng ô trong ma trận ta gọi là phần tử. Ví dụ ta cho ma trận 2 hàng và 3 cột như sau:

$$A = \begin{bmatrix} 1 & 2 & 3 \\ 6 & 5 & 4 \end{bmatrix}$$

Ta đánh số phần tử của ma trận bằng chỉ số hàng và chỉ số cột $A[i][j]$, trong đó i là chỉ số hàng và j là chỉ số cột. Chẳng hạn với ma trận trên ta có $A[1][1] = 1, A[1][2] = 2, A[1][3] = 3, A[2][1] = 6, A[2][2] = 5, A[2][3] = 4$.

Khi ma trận chỉ có 1 hàng, ta có ma trận hàng như sau: $[a_1 \quad \cdots \quad a_n]$ và thường gọi là vector hàng.

Khi ma trận chỉ có 1 cột, ta có ma trận cột như sau: $\begin{bmatrix} a_1 \\ \cdots \\ a_n \end{bmatrix}$ và ta gọi là vector cột.

Khi số hàng và số cột bằng nhau ta có khái niệm ma trận vuông. Chẳng hạn ma trận vuông cấp 3 như sau: $\begin{bmatrix} 1 & 2 & 0 \\ 6 & 1 & 0 \\ 8 & 0 & 1 \end{bmatrix}$.

Trong tin học lập trình ta mở rộng khái niệm ma trận, phần tử của ma trận không chỉ là số thực mà còn có thể là ký tự hay là một kiểu dữ liệu khác. Ví dụ sau đây là một ma trận ký tự: $\begin{bmatrix} a & b & c \\ x & y & z \\ a & s & d \end{bmatrix}$.

Ngôn ngữ lập trình C^{++} cung cấp kiểu dữ liệu mảng hai chiều để biểu diễn ma trận. Ta chú ý rằng kiểu dữ liệu này rất quan trọng trong nhiều lĩnh vực nghiên cứu như xử lý ảnh, thống kê, trí tuệ nhân tạo…

2. Khai báo mảng hai chiều

Cú pháp khai báo mảng như sau:

```
<tên kiểu dữ liệu> <tên mảng>[số_pt_hàng][số_pt cột];
```

Trong đó `<tên kiểu dữ liệu>` là kiểu nguyên int, long long hoặc kiểu thực float hay kiểu ký tự…Ví dụ ta khai báo: `int a[100][100];` sẽ cho ta 100×100 phần tử từ $a[0][0]$ đến $a[99][99]$.

3. Cách sử dụng

Sau khi khai báo mảng ta truy cập phần tử thông qua chỉ số của phần tử như sau: `<tên_mảng> [chỉ_số_hàng][chỉ số cột]`, ví dụ muốn truy cập phần tử hàng 5 cột 6 của mảng `a` ta viết `a[5][6]`.

Để đọc dữ liệu cho mảng ta kết hợp hai vòng lặp lồng nhau để đọc cho từng phần tử như ví dụ sau:

```
for(int i=1;i<=10;i++)
    for(int i=1;i<=10;i++)
        cin>>a[i][j];
```

Để in dữ liệu mảng ta kết hợp hai vòng lặp lồng nhau để in cho từng phần tử như ví dụ sau:

```
for(int i=1;i<=10;i++)
   {
      for(int i=1;i<=10;i++)
          cout<<a[i][j];
      cout<<endl;//lệnh này để ngắt dòng
    }
```

Muốn tác động tới từng phần tử của mảng ta truy cập qua chỉ số như ví dụ sau:

```
for(int i=1;i<=10;i++)
    for(int i=1;i<=10;i++)
        S = S + a[i][j];
```

B. CÁC VÍ DỤ MẪU

Ví dụ 5.1: Ma trận là một bảng số gồm m hàng, n cột. Khi số hàng bằng số cột ta có khái niệm ma trận vuông. Ta sử dụng cấu trúc mảng hai chiều để biểu diễn ma trận. Chương trình sau nhập vào một ma trận có m hàng n cột và in ra tổng tất cả các phần tử trong ma trận.

```
#include <bits/stdc++.h>
using namespace std;
int a[105][105], m,  n, S = 0;
int main()
{
    cin>>m>>n;
    for(int i = 1; i<=m;i++)
        for(int j = 1; j<=n; j++)
            {
                cin>>a[i][j];
                S = S + a[i][j];
            }
    cout<<S;
}
```

Ví dụ 5.2: Chương trình sau nhập vào một ma trận vuông và in ra các phần tử trên đường chéo chính.

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

```
using namespace std;
int a[105][105], n, S = 0;
int main()
{
    cin>>n;
    for(int i = 1; i<=n;i++)
        for(int j = 1; j<=n; j++)
            {
                cin>>a[i][j];
            }
    for(int i = 1; i<=n;i++)
            cout<<a[i][j]<< " ";
}
```

C. BÀI TẬP ÁP DỤNG

Bài 5.1 – (N0501A) In ma trận

Yêu cầu: Viết chương trình nhập vào một ma trận vuông. In ra chính ma trận vừa nhập.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm $n (n \leq 100)$,
- n dòng sau, dòng thứ i ghi n số nguyên biểu diễn hàng thứ i của ma trận.

Kết quả: In ra chính ma trận vừa nhập.

Ví dụ:

input	output
2 1 1 2 2	1 1 2 2

Bài 5.2 – (N0502A) Tổng trên ma trận

Yêu cầu: Viết chương trình nhập vào một ma trận vuông. In ra chính tổng các phần tử trong ma trận vuông vừa nhập.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm $n (n \leq 100)$,
- n dòng sau, dòng thứ i ghi n số nguyên biểu diễn hàng thứ i của ma trận.

Kết quả: In ra tổng các phần tử trong ma trận.

Ví dụ:

input	output
2 1 1 2 2	6

Bài 5.3 – (N0503B) Hàng có tổng lớn nhất

Yêu cầu: Viết chương trình nhập vào một ma trận vuông. In ra tổng lớn nhất của một hàng trong ma trận đã cho.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm $n (n \leq 100)$,
- n dòng sau, dòng thứ i ghi n số nguyên biểu diễn hàng thứ i của ma trận.

Kết quả: In ra tổng lớn nhất của một hàng trong ma trận.

Ví dụ:

input	output
3 1 1 1 2 2 3 4 4 4	12

Bài 5.4 – (N0504B) Cột có tổng lớn nhất

Yêu cầu: Viết chương trình nhập vào một ma trận vuông. In ra tổng lớn nhất của một cột trong ma trận đã cho.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm $n (n \leq 100)$,
- n dòng sau, dòng thứ i ghi n số nguyên biểu diễn hàng thứ i của ma trận.

Kết quả: In ra tổng lớn nhất của một cột trong ma trận.

Ví dụ:

input	output
3 1 1 1 2 2 3 4 4 4	8

Bài 5.5 – (N0505A) Giá trị chẵn lớn nhất

Yêu cầu: Viết chương trình nhập vào một ma trận vuông. In ra giá trị chẵn lớn nhất trong ma trận.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm $n (n \leq 100)$,
- n dòng sau, dòng thứ i ghi n số nguyên biểu diễn hàng thứ i của ma trận.

Kết quả: In ra giá trị chẵn lớn nhất trong ma trận.

Ví dụ:

input	output
3 1 1 1 2 2 3 4 4 6	6

Bài 5.6 – (N0506A) Tổng trên đường chéo chính

Yêu cầu: Viết chương trình nhập vào một ma trận vuông. In ra tổng các phần tử trên đường chéo chính của ma trận, đường chéo chính là các phần tử có chỉ số hàng bằng chỉ số cột.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm $n (n \leq 100)$,
- n dòng sau, dòng thứ i ghi n số nguyên biểu diễn hàng thứ i của ma trận.

Kết quả: In ra tổng các phần tử trên đường chéo chính.

Ví dụ:

input	output
3 1 1 1 2 2 3 4 4 6	9

Bài 5.7 – (N0507B) Tổng trên đường chéo phụ

Yêu cầu: Viết chương trình nhập vào một ma trận vuông. Tính tổng các phần tử lẻ trên đường chéo phụ, phần tử thuộc đường chéo phụ có tổng chỉ số hàng và chỉ số cột bằng $n + 1$ với n là cỡ ma trận.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm $n (n \leq 100)$,
- n dòng sau, dòng thứ i ghi n số nguyên biểu diễn hàng thứ i của ma trận.

Kết quả: In ra tổng các phần tử lẻ trên đường chéo phụ.

Ví dụ:

input	output
3 1 1 1 2 2 3 3 4 6	4

Bài 5.8 – (N0508C) Tổng trên biên ma trận

Yêu cầu: Viết chương trình nhập vào một ma trận vuông. Tính tổng các phần tử nằm trên biên của ma trận.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm $n (n \leq 100)$,
- n dòng sau, dòng thứ i ghi n số nguyên biểu diễn hàng thứ i của ma trận.

Kết quả: In ra tổng các phần tử lnằm trên biên của ma trận. Chẳng hạn với ma trận $M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, tổng trên biên của ma trận M là $1 + 2 + 3 + 1 + 1 + 1 + 4 = 14$.

Ví dụ:

Input	output
3 1 2 3 4 5 1 1 1 1	14

Bài 5.9 – (N0509A) Tổng hai ma trận

Yêu cầu: Cho hai ma trận cùng cỡ, việc tính tổng hai ma trận chỉ đơn giản là lấy tương ứng các phần tử của hai ma trận cộng cho nhau. Ví dụ ma trận $C = A + B$ có $c_{ij} = a_{ij} + b_{ij}$. Hãy viết chương trình cộng hai ma trận với nhau.

Dữ liệu:

- Dòng đầu tiên ghi 2 số nguyên không âm m, n $(m, n \leq 100)$,
- m dòng tiếp, mỗi dòng ghi n số nguyên biểu diễn hàng thứ i của ma trận A $(0 < |a_{ij}| \leq 10^9)$,
- m dòng tiếp, mỗi dòng ghi n số nguyên biểu diễn hàng thứ i của ma trận B $(0 < |a_{ij}| \leq 10^9)$

Kết quả: m dòng, mỗi dòng ghi n số biểu diễn ma trận tổng.

Ví dụ:

input	output

2 2 1 1 1 1 2 3 2 3	3 4 3 4

Bài 5.10 – (N0510B) Tích hai ma trận

Yêu cầu: So với phép toán cộng ma trận thì phép toán tích ma trận có sự phức tạp hơn rất nhiều. Cho hai ma trận A có cỡ $m \times n$ và ma trận B có cỡ $n \times p$. Khi đó ta mới thực hiện được phép nhân hai ma trận $C = A \times B$ như sau:

- Ma trận C sẽ có cỡ $m \times p$,
- Phần tử c_{ij} của ma trận C được tính bởi công thức:
  $c_{ij} = a_{i1} \times b_{1j} + a_{i2} \times b_{2j} + \ldots + a_{in} \times b_{nj}$.

Chú ý rằng phép nhân ma trận không có tính giao hoán, nghĩa là $A \times B \neq B \times A$.

Ví dụ: Cho $A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \end{bmatrix}, B = \begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 0 & 1 \end{bmatrix} \Rightarrow C = A \times B = \begin{bmatrix} 2 & 2 \\ 3 & 3 \end{bmatrix}$

Dữ liệu:

- Dòng đầu tiên ghi 3 số nguyên không âm m, n, p $(m, n, p \leq 100)$,
- m dòng tiếp, mỗi dòng ghi n số nguyên biểu diễn hàng thứ i của ma trận A $(0 < |a_{ij}| \leq 10^9)$,
- n dòng tiếp, mỗi dòng ghi p số nguyên biểu diễn hàng thứ i của ma trận B $(0 < |a_{ij}| \leq 10^9)$,

Kết quả: m dòng, mỗi dòng ghi p số biểu diễn ma trận tích $C = A \times B$.

Ví dụ:

input	output
2 3 2 1 1 1 1 2 1 1 0 1 1 0 1	2 2 3 3

Bài 5.11 – (N0511C) Tổng các bảng vuông

Yêu cầu: Ta xem xét một bảng số vuông, còn gọi là ma trận vuông cỡ $n \times n$. Xét các hình vuông đồng tâm có tâm là tâm của bảng số ban đầu. Hãy tính tổng các số có trên các cạnh của mỗi hình vuông đó. Ví dụ với n là số lẻ thì tâm của ma

trận là phần tử $a[n/2+1; n/2+1]$, còn với n chẵn thì tâm được xem là giao điểm của đường chéo chính và đường chéo phụ.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm $n (n \leq 300)$,
- n dòng sau, mỗi dòng ghi n số nguyên biểu diễn hàng thứ i của ma trận (các số nguyên đều có giá trị tuyệt đối nhỏ hơn 10^6).

Kết quả: một dòng gồm các số lần lượt là tổng các số trên cạnh của những hình vuông đồng tâm với hình vuông ban đầu *(từ hình vuông có kích thước nhỏ nhất tới hình vuông có kích thước lớn nhất $n \times n$)*

Ví dụ: ma trận $M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 1 \\ 1 & 1 & 1 \end{bmatrix}$, có hai bảng vuông tương ứng có kích thước 1×1 và 3×3 với các tổng là 5 và $1 + 2 + 3 + 1 + 1 + 1 + 1 + 4 = 14$.

Ví dụ:

input	output
4 1 2 3 1 4 5 6 1 1 2 3 1 1 1 1 1	16 18

CHƯƠNG 6 – KIỂU DỮ LIỆU XÂU KÝ TỰ

A. KIẾN THỨC GHI NHỚ

1. Hai kiểu xâu trong C++

Trong C^{++} có 2 kiểu xâu. Một là kiểu C – style, đây bản chất là mảng các ký tự. Hai là kiểu string – C^{++} – style, đây là một kiểu dữ liệu dạng vector với nhiều tiện lợi. Trong hai kiểu thì kiểu string có nhiều ưu điểm và các phiên bản mới nhất luôn có những cập nhật cho kiểu này. Trong tài liệu này ta chỉ đề cập tới kiểu xâu ký tự - string.

2. Khai báo và sử dụng

Cú pháp khai báo của kiểu string:

```
string <tên xâu>;
```

Đọc dữ liệu xâu ta có 2 câu lệnh sau:

```
cin >> <tên xâu>;
```

hoặc:

```
getline(cin, <tên xâu>);
```

Lệnh thứ nhất sẽ đọc xâu tới dấu trắng đầu tiên và dừng lại, có nghĩa là lệnh này sẽ không đọc được xâu có ký tự cách trống. Lệnh thứ hai sẽ khắc phục được việc không đọc được ký tự trống như lệnh thứ nhất.

Về bản chất xâu ký tự vẫn là các ký tự đặt cạnh nhau và truy cập vào các ký tự thông qua chỉ số. Ví dụ, để truy cập tới phần tử thứ i của xâu S ta viết S[i].

Ta chú ý rằng xâu trong C^{++} được đánh số từ 0.

3. Các phép toán và hàm thành viên

- Hàm quản lý số lượng ký tự `length()` hoặc `size()`.

```
cout<<s.length(); //in ra độ dài xâu
cout <<s.size();//cũng in ra độ dài xâu
```

- Hàm $s.front()$ trả về phần tử đầu tiên.
- Hàm $s.back()$ trả về phần tử cuối cùng.
- Hàm $s.substr(k,l)$ trích xuất xâu con từ vị trí k với số ký tự l.
- Phương thức chèn xâu $S1$ và xâu $S2$ với cú pháp $S1.insert(vị\ trí, S2)$
- Phương thức $s.erase(k,l)$ xóa xâu s từ vị trí k đi l ký tự.
- Hàm $S2.find(S1)$ để tìm kiếm xâu S1 trong xâu S2.

B. CÁC VÍ DỤ MẪU

Ví dụ 6.1: Chương trình sau nhập vào một xâu ký tự và biến các ký tự của xâu thành ký tự hoa rồi in ra xâu mới.

```
#include <bits/stdc++.h>
#define ll long long
```

```
using namespace std;
string S
int main()
{
    getline(cin,S);
    S = " " + S;
    for(int i = 1; i<S.size();i++)
        S[i] = toupper(S[i]);
    cout<<S;
}
```

Ví dụ 6.2: Chương trình sau nhập vào hai xâu ký tự và ghép hai xâu ký tự thành một xâu.

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
string S1,S2
int main()
{
    getline(cin,S1);
    getline(cin,S2);
    string S = S2 + S1;
    cout<<S;
}
```

Ví dụ 6.3: Chương trình sau nhập vào một xâu ký tự bao gồm cả dấu cách trống, đếm và in ra các dấu cách trống trong xâu.

```
#include <bits/stdc++.h>
#define ll long long
using namespace std;
string S;
int dem = 0;
int main()
{
    getline(cin,S);
    for(int i = 0;i<S.size();i++)
        if(S[i]==' ') dem++;
    cout<<dem;
}
```

C. BÀI TẬP ÁP DỤNG

Bài 6.1 – (N0601A) Độ dài xâu

Yêu cầu: In ra độ dài của xâu ký tự S. Trong kiểu string có 2 hàm thành viên quản lý độ dài xâu là `S.length()` và `S.size()`.

Dữ liệu: Một chuỗi kí tự S có độ dài bất kỳ.

Kết quả: Là độ dài của chuỗi ký tự S trên.

Ví dụ:

input	output
abcd	4

Bài 6.2 – (N0602A) Đếm ký tự

Yêu cầu: Viết chương trình đếm số ký tự Ch xuất hiện trong xâu S.

Dữ liệu:

- Dòng 1 là chuỗi kí tự S có độ dài nhỏ 100 ký tự,
- Dòng 2 là ký tự Ch.

Kết quả: Là số lượng ký tự Ch trong xâu S.

Ví dụ:

input	output
bcdaaavab a	4

Bài 6.3 – (N0603A) Ký tự hoa

Yêu cầu: Viết chương trình làm hoa các ký tự trong xâu S cho trước.

Dữ liệu: Một dòng là chuỗi kí tự S có độ dài nhỏ hơn 1000 ký tự bao gồm các ký tự la tinh thường và hoa.

Kết quả: Là xâu S sau khi đã được làm hoa các ký tự.

Ví dụ:

input	output
abcd	ABCD

Bài 6.4 – (N0604A) Ký tự số

Yêu cầu: Viết chương trình đếm các ký tự chữ số trong xâu S cho trước.

Dữ liệu: Một dòng là chuỗi kí tự S có độ dài nhỏ hơn 1000 ký tự bao gồm các ký tự la tinh và các chữ số.

Kết quả: Là số lượng chữ số trong chuỗi ký tự S.

Ví dụ:

input	output
abcd123abde	3

Bài 6.5 – (N0605B) Xâu đối xứng

Yêu cầu: Một xâu đối xứng là xâu mà khi viết ngược lại thì ta được xâu ban đầu. Ví dụ xâu ABBA là một xâu đối xứng. Hãy viết chương trình kiểm tra tính đối xứng của một xâu.

Dữ liệu: Một dòng là xâu kí tự S có độ dài nhỏ hơn 1000 ký tự bao gồm các ký tự la tinh.

Kết quả: Ghi YES nếu S là xâu đối xứng, ghi NO nếu S là xâu không đối xứng.

Ví dụ:

input	output
ABBA	YES

Bài 6.6 – (N0606B) Tổng chữ số

Yêu cầu: Cho một số nguyên dương X. Yêu cầu: Hãy tính tổng các chữ số của số nguyên dương X.

Dữ liệu: Một dòng là số nguyên dương X ($X < 10^{100}$).

Kết quả: Ghi tổng các chữ số của X.

Ví dụ:

input	output
1234	10

Bài 6.7 – (N0607B) Đếm số từ

Yêu cầu: Viết chương trình nhập vào một văn bản là một chuỗi kí tự và có thể chứa các kí tự cách trống. Đếm số từ trong chuỗi ký tự đó.

Dữ liệu: Một dòng là chuỗi ký tự S ($0 < length(S) \leq 1000$).

Kết quả: Số từ trong chuỗi S.

Ví dụ:

input	output
Cong hoa xa hoi	4

Bài 6.8 – (N0608B) Loại bỏ chữ số

Yêu cầu: Viết chương trình nhập vào một văn bản là một chuỗi kí tự bao gồm chữ cái và chữ số. Loại bỏ các chữ số trong chuỗi ký tự đó.

Dữ liệu: Một dòng là chuỗi ký tự S ($0 < length(S) \leq 1000$).

Kết quả: Chuỗi S sau khi đã loại bỏ các ký tự số.

Ví dụ:

input	output
ABCD1234abcd	ABCDabcd

Bài 6.9 – (N0609B) Số ký tự phân biệt

Yêu cầu: Viết chương trình nhập vào một văn bản là một chuỗi kí tự bao gồm chữ cái la tinh thường. Hãy in ra số lượng các ký tự phân biệt trong chuỗi.

Dữ liệu: Một dòng là chuỗi ký tự S $(0 < length(S) \leq 2000)$.

Kết quả: Số lượng các ký tự phân biệt trong chuỗi ký tự S.

Ví dụ:

input	output
abca	3

Bài 6.10 – (N0610C) Mã hóa 1

Yêu cầu: Mr X cần gửi một văn bản quan trọng tới các coder của mình. Để đảm bảo tính bảo mật của thông tin được ghi trong văn bản, ông quyết định sẽ mã hóa văn bản trước khi gửi. Văn bản là một xâu S gồm các chữ cái latinh thường. Ông ấy chia đoạn văn bản thành hai đoạn liên tiếp nhau là Sa và Sb. Lần lượt viết các xâu Sa và Sb theo thứ tự ngược lại ta nhận được xâu mã hóa Q. Ví dụ, nội dung bức thư S = 'programming' với với khóa $k = 7$ sẽ được chia thành 2 đoạn: Sa = 'program', Sb = 'ming', nhận được xâu mã hóa Q = 'margorpgnim'. Để cho coder của mình có thể hiểu được nội dung bức thư ông ấy đã gửi kèm theo xâu mã hóa Q là một số nguyên dương k cho biết độ dài xâu Sa. Bạn hãy giải mã bức thư này.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm k.
- Dòng thứ hai ghi xâu mã hóa Q; độ dài của xâu Q không vượt quá 10^6

Kết quả: Đưa ra xâu kí tự sau khi được giải mã.

Ví dụ:

input	output
7 margorpgnim	programming

Bài 6.11 – (N0611C) Mã hóa 2

Yêu cầu: Trước công nguyên, nhà quân sự người La Mã Julius Ceasar đã nghĩ ra phương pháp mã hóa một bản tin như sau: thay thế mỗi chữ cái trong bản tin bằng chữ cái đứng sau nó k vị trí trong bảng chữ cái. Giả sử với $k = 3$, ta có bảng chuyển đổi như sau:

Chữ ban đầu: a b c d e f g h i j k l m n o p q r s t u v w x y z

Chữ thay thế: d e f g h i j k l m n o p q r s t u v w x y z a b c

Khi đó bản tin: ‘attack’ sau khi mã hóa sẽ có bản mã ‘dwwdfn’ và sau đó Ceasar gửi bản mã cho cấp dưới của mình.

Nhận được bản mã và khóa k, cấp dưới của ông sẽ phải giải mã bản tin để đọc nội dung của nó.

Dữ liệu:

- Dòng đầu tiên ghi khóa k,
- Dòng thứ 2 ghi bản tin sau khi đã được mã hóa.

Kết quả: Đưa ra bản tin ở trạng thái chưa được mã hóa.

Giới hạn: $1 \leq k \leq 25$, độ dài bản tin không vượt quá 100 kí tự.

Ví dụ:

input	output
7 wyvnyhttpun	programming

Bài 6.12 – (N0612D) Mã hóa 3

Yêu cầu: Trong bảng mã ASCII, 26 kí tự chữ cái thường từ ‘a’ đến ‘z’ được mã hóa tương ứng bằng các số tự nhiên từ 97 đến 122. Cho một xâu kí tự S chỉ chứa toàn các kí tự chữ cái thường. Gọi P là xâu mã hóa tương ứng của xâu S bằng cách mã hóa từng ký tự trong S (theo bảng mã ASCII) và viết liên tiếp nhau. Ví dụ: S = ‘ab’ thì P = ‘9798’. Hãy viết chương trình nhập vào từ bàn phím một xâu đã mã hóa P (có không quá 255 kí tự) và in ra màn hình xâu kí tự S.

Dữ liệu: Chứa một xâu đã mã hóa P.

Kết quả: In ra màn hình xâu kí tự S.

Ví dụ:

input	output
979899	abc

CHƯƠNG 7 - KIỂU DỮ LIỆU TỆP VĂN BẢN

A. KIẾN THỨC GHI NHỚ

1. Lệnh đồng bộ tệp và vào ra chuẩn

Bản chất vào ra chuẩn cũng là tương tác trực tiếp với tệp bộ nhớ đệm bàn phím và tệp bộ nhớ đệm màn hình. Ta có các lệnh đồng bộ việc đọc vào từ tệp với việc đọc vào từ bàn phím như sau

```
freopen("Tentep.inp","r",stdin);
```

Tương tự ta có các lệnh đồng bộ việc ghi ra màn hình và file dữ liệu như sau:

```
freopen("Tentep.out","w",stdout);
```

Sau khi có các lệnh đồng bộ này ta đọc dữ liệu từ bàn phím sẽ tương đương với đọc từ tệp và ghi dữ liệu ra màn hình tương đương với ghi dữ liệu ra tệp.

2. Tệp văn bản trong C++ với thư viện fstream

Ngoài việc đồng bộ như trên ta có thể sử dụng thư viện fstream để đọc và ghi dữ liệu với tệp như sau:

- Đọc dữ liệu vào với lệnh ifsream như sau:

```
ifstream in1("file.INP");
in1>>a;//Đọc dữ liệu từ tệp
in1.close();
```

- Ghi dữ liệu ra với lệnh ofsream như sau:

```
ofstream out1("file.OUT");
out1<<a;//Ghi dữ liệu vào tệp
out1.close();
```

B. CÁC VÍ DỤ MẪU

Ví dụ 7.1: Chương trình sau mở file văn bản có tên BT.INP và đọc hai số nguyên a, b từ đó và ghi vào file văn bản BT.OUT tổng của hai số $a + b$.

```
#include <bits/stdc++.h>
using namespace std;
long long n, dem = 0;
int main()
{
    freopen("BT.INP","r",stdin);
    freopen("BT.OUT","w",stdout);
    long long a,b;
    cin>>a>>b;
    cout<<a+b;
}
```

Chú ý: Trong các kỳ thi lập trình thuật toán thường chỉ mở file để đọc và ghi một lần nên cách làm trên là ngắn gọn và đủ dùng. Ta có thể thay tên file bằng đường

dẫn tuyệt đối, còn nếu không thì các file phải được đặt trong thư mục chứa chương trình.

Ví dụ 7.2: Chương trình sau mở một file văn bản BT1.INP chứa số nguyên a, sau đó đóng file và mở file văn bản BT2.INP chứa số nguyên b, đóng file và ghi kết quả a*b vào file văn bản BT3.OUT. Trong chương trình sau ta sử dụng các câu lệnh `ifstream` để hướng luồng vào tới file cần vào dữ liệu vào câu lệnh `ofstream` để hướng luồng ra tới file ra dữ liệu.

```
#include <bits/stdc++.h>
using namespace std;
long long n, dem = 0;
int main()
{
    long long a,b;
    ifstream in1("BT1.INP");
    in1>>a;
    in1.close();
    ifstream in2("BT2.INP");
    in2>>b;
    in2.close();
    ofstream out("BT3.OUT");
    out<<a+b;
}
```

C. BÀI TẬP ÁP DỤNG

Bài 7.1 – (N0701A) Số âm

Yêu cầu: Viết chương trình nhập các số nguyên từ file "BT.INP" cho đến khi gặp số nguyên âm đầu tiên, in số dương trước số âm đầu tiên đó vào file "BT.OUT".

Dữ liệu: vào từ file BT.INP

- Một dòng ghi các số nguyên 32 bit.

Kết quả: ghi ra file BT.OUT

- Một dòng ghi số dương trước số âm đầu tiên.

Ví dụ:

BT.INP	BT.OUT
5 11 22 33 44 55	30

Bài 7.2 – (N0702B) Min max ra hai file

Yêu cầu: Viết chương trình đọc một dãy số nguyên từ file văn bản "SO.INP". Tìm và in ra phần tử lớn nhất vào file SO.OU1 đồng thời tìm và in ra phần tử nhỏ nhất vào file SO.OU2.

Dữ liệu: file văn bản SO.INP

- Dòng đầu tiên ghi số nguyên không âm $n\,(0<n\le 100)$,
- Dòng thứ 2 ghi n số nguyên dương kiểu 64 – bit.

Kết quả:

- In ra **file văn bản SO.OU1** phần tử lớn nhất.
- In ra **file văn bản SO.OU2** phần tử nhỏ nhất.

Ví dụ:

SO.INP	SO.OU1	SO.OU2
3 1 4 7	7	1

Bài 7.3 – (N0703B) Sắp xếp dữ liệu ngoài

Yêu cầu: Viết chương trình đọc một dãy số nguyên từ file văn bản "SORT.IN1" và một dãy số nguyên từ file văn bản "SORT.IN2". Trộn hai dãy số lại sau đó sắp xếp và in ra file văn bản "SORT.OUT".

Dữ liệu 1: file văn bản SORT.IN1

- Dòng đầu tiên ghi số nguyên không âm m $(0 < n \le 100)$,
- Dòng thứ 2 ghi m số nguyên dương kiểu 64 – bit.

Dữ liệu 2: file văn bản SORT.IN2

- Dòng đầu tiên ghi số nguyên không âm n $(0 < n \le 100)$,
- Dòng thứ 2 ghi n số nguyên dương kiểu 64 – bit.

Kết quả:

- In ra file văn bản SORT.OUT dãy sau khi trộn và sắp xếp.

Ví dụ:

SORT.IN1	SORT.IN1	SORT.OUT
3 1 4 7	4 2 3 5 9	1 2 3 5 7 9

Bài 7.4 – (N0703B) Đọc file không biết số lượng

Yêu cầu: Viết chương trình đọc nhiều số nguyên trên nhiều dòng từ file văn bản "SUM.INP". File gồm nhiều dòng, mỗi dòng là một dãy gồm nhiều số nguyên không xác định. In ra file văn bản "SUM.OUT" gồm số dòng tương ứng, mỗi

dòng là một số nguyên là tổng số của các số nguyên trên dòng dữ liệu từ file đầu vào.

Dữ liệu: file văn bản SUM.INP

- Gồm nhiều dòng, mỗi dòng ghi nhiều số nguyên kiểu 32 bit
- Số lượng dòng không quá 100.

Kết quả: file văn bản SUM.INP

- Gồm nhiều dòng tương ứng, mỗi dòng là tổng của các số nguyên tương ứng trong file đầu vào.

Ví dụ:

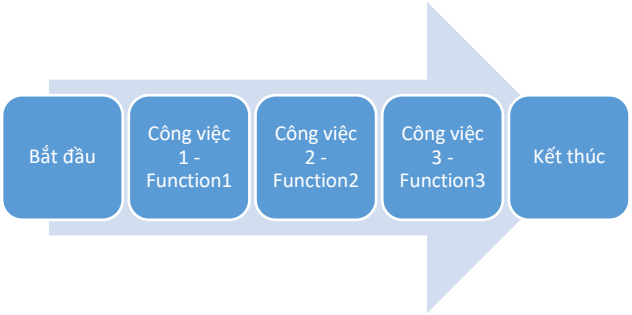
SUM.INP	SUM.OUT	SUM.INP	SUM.OUT
1 2 3 1 4 1 2 3 6	6 5 12	1 2 5 6 1 2	14 3

CHƯƠNG 8 - HÀM VÀ CẤU TRÚC HÀM

A. KIẾN THỨC GHI NHỚ

1. Khái niệm về hàm

Hàm trong ngôn ngữ lập trình gọi là function. Nó là một phần không thể thiếu của ngôn ngữ lập trình. Có nhiều định nghĩa khác nhau về hàm, sau đây là một cách hiểu: "Function là một đoạn các câu lệnh có thể tái sử dụng. Function cho phép lập trình viên cấu trúc chương trình thành những phân đoạn khác nhau để thực hiện những công việc khác nhau". Ta có thể hình dung một chương trình là một công việc, công việc đó có thể phân ra thành nhiều công việc nhỏ, mỗi công việc nhỏ như vậy ta có thể chia nhỏ hơn các nhiệm vụ. Trong các nhiệm vụ và công việc như vậy ta thiết kế thành các hàm thực hiện. Một mô hình đơn giản nhất về hàm như sau:



Function (hàm) được người ta ví như một cái hộp đen, chúng ta không biết bên trong nó là gì, nhưng "hộp đen" có một đầu vào (input) và một đầu ra (output). Việc của chúng ta khi sử dụng cái "hộp đen" này (thực hiện lời gọi hàm) là đưa những dữ liệu đầu vào tương thích vào đầu vào của nó, và nó sẽ cho kết quả tại đầu ra.

2. Khai báo và sử dụng hàm

Để sử dụng hàm thì ta phải khai báo hàm. Ta thường có hai cách khai báo hàm như sau. Cách thứ nhất ta khai báo hàm trên chương trình chính trong đó cài đặt hàm và sử dụng trong thân chương trình chính như ví dụ sau:

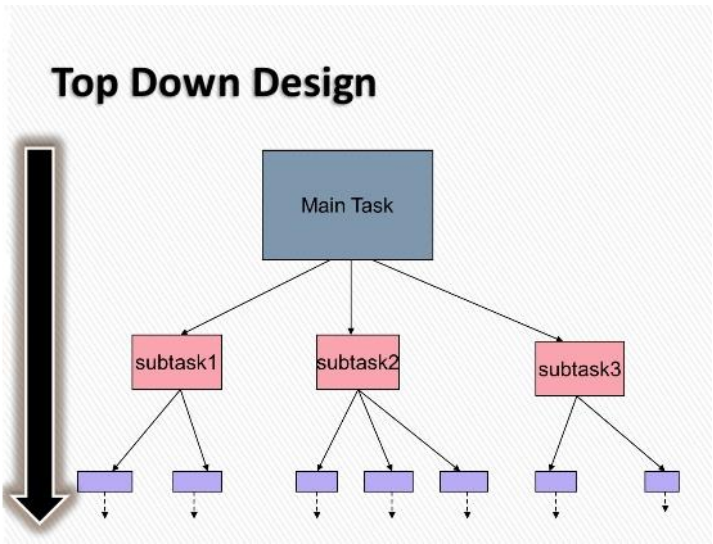
```
#include <bits/stdc++.h>
#define nmax 100
using namespace std;
long long sum(long long a, long long b)
{
    return a + b;
}
long long x = 2, y = 10
int main()
{
    cout<<sum(x,y);
}
```

Trong ví dụ trên ta khai báo một hàm sum tác động tới hai tham số a, b và cho kết quả là tổng của hai số a, b. Với hàm sum, trong chương trình chính ta thực hiện lệnh gọi tổng hai số x, y.

Bên cạnh cách thứ nhất, ta có thể sử dụng cách thứ hai để khai báo hàm nguyên mẫu như sau:

```
#include <bits/stdc++.h>
#define nmax 100
using namespace std;
long long sum(long long a, long long b);
long long x,y;
int main()
{
    cin>>x>>y;
    cout<<sum(x,y);
}
long long sum(long long a, long long b)
{
    return a+b;
}
```

Cách này phù hợp với việc khai báo hàm trước, viết hàm sau với tư duy phân hoạch công việc từ tổng quát cho tới chi tiết – là một đặc trưng của hướng tiếp cận top – down khi giải quyết một vấn đề.



4. Biến địa phương và biến toàn cục

Các biến được khai báo trong các hàm được gọi là biến địa phương. Biến địa phương chỉ có tầm ảnh hưởng trong chính hàm đó. Khi hàm kết thúc, các biến địa phương sẽ được giải phóng và phá hủy. Ngôn ngữ lập trình C++ xem tất cả mọi đoạn chương trình thực hiện đều là hàm, kể cả chương trình chính `main()` cũng

là một hàm, ta gọi là hàm chính; các vòng lặp for cũng xem như là một hàm, các biến chạy trong vòng lặp sẽ giải phóng khi ra khỏi vòng lặp.

Các biến được khai báo ở trên cùng có ảnh hưởng tới tất cả các đoạn chương trình ta gọi là biến toàn cục. Biến toàn cục có thể sử dụng ở bất cứ đâu, tuy nhiên nếu nó đặt tên cùng với tên của một biến địa phương thì sẽ xảy ra hiện tượng "phép vua thua lệ làng", khi vào trong hàm sẽ ưu tiên biến địa phương.Vì vậy khi đặt tên biến cần tránh tình trạng đặt tên trùng. Chúng ta lưu ý không khai báo nhiều biến để chiếm bộ nhớ trong hàm, vì ngăn xếp chương trình con của C^{++} có dung lượng hữu hạn. Vì hàm `main()` cũng là một hàm nên chúng ta không được khai báo các mảng cỡ lớn trong thân hàm `main()`.

5. Tham số - tham biến, tham trị

Khi xây dựng hàm, thông thường hàm sẽ có các đối số để tác độ vào, ta gọi chung là tham số. Tham số khi nằm trong hàm mà chưa có lời gọi hàm thì ta gọi là tham số hình thức. Còn khi lời gọi hàm được thực hiện, tham số hình thức sẽ được truyền cụ thể các biến số và khi đó các tham số được gọi là tham số thực sự. Ta xem xét lại ví dụ trên:

```
#include <bits/stdc++.h>
#define nmax 100
using namespace std;
long long sum(long long a, long long b)
//Tham số a, b là tham số hình thức
{
    return a + b;
}
long long x = 2, y = 10
int main()
{
    cout<<sum(x,y);
//Lời gọi hàm được thực hiện thì x, y là
//tham số thực sự
}
```

Khi có lời gọi hàm thì các tham số thực sự sẽ được truyền vào hàm và các tham số hình thức sẽ được thay thế bởi các tham số thực sự.

Khi ra khỏi hàm, các tham số thực sự sẽ có hai tình huống. Tình huống thứ nhất là trong thân hàm không có những lệnh làm thay đổi giá trị của tham số thực sự. Với tình huống này không có gì phải nói, vì như thế các đối số sau khi tham gia vào chương trình con sẽ trở lại tham gia vào các đoạn chương trình khác. Tình huống thứ hai là, trong thân hàm có những lệnh làm thay đổi giá trị của tham số thực sự. Nếu khi ra khỏi hàm, kết quả không được bảo toàn mà trả về giá trị của

tham số thực sự trước khi tham gia hàm thì ta có khái niệm tham trị. Cụ thể ta xét đoạn chương trình sau:

```
#include <bits/stdc++.h>
#define nmax 100
using namespace std;
long long doicho(long long a, long long b)
{
    long long tam = a;
    a = b;
    b = tam;
}
long long x,y;
int main()
{
    long long x = 1, y = 2;
    doicho(x,y);
    cout<<x<<" "<<y;
}
Output: 1 2
```

Trong đoạn chương trình trên dù rằng trong hàm doicho () đã thực hiện đổi chỗ 2 số a, b tuy nhiên khi in ra kết quả không cho ta điều này vì các tham số được khai báo trong hàm doicho () là các tham trị.

Để lưu giữ lại sự thay đổi của các tham số, ta có khái niệm tham biến. Tham biến là tham số mà sau khi thực thi hàm thì kết quả các phép toán tác động lên tham số sẽ được bảo toàn. Cụ thể ta xét đoạn chương trình sau:

```
#include <bits/stdc++.h>
#define nmax 100
using namespace std;
long long doicho(long long &a, long long &b)
//Thêm các ký tự & trước a, b
{
    long long tam = a;
    a = b;
    b = tam;
}
long long x,y;
int main()
{
    long long x = 1, y = 2;
    doicho(x,y);
```

```
    cout<<x<<" "<<y;
}
Output: 2 1
```

Như vậy để cài đặt tham biến ta cần thêm ký hiệu & vào trước danh sách tham số. Bản chất ở đây ta đang sử dụng con trỏ để trỏ vào địa chỉ của tham số.

Từ trên ta có kết luận sau:

- Dùng tham trị khi bạn không muốn giá trị của biến bị thay đổi.
- Dùng tham biến khi bạn muốn hàm thay đổi giá trị biến của bạn.
- Dùng tham biến sẽ tiết kiệm bộ nhớ hơn.
- Nếu bạn dùng tham biến, 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 ta phải truyền theo tham biến bởi bản chất tên mảng chính là địa chỉ của mảng.

Xét ví dụ tham biến là mảng như sau:

```
#include <bits/stdc++.h>
#define nmax 100
using namespace std;
long long sum(long long x[], int x_size)
{
    long long temp = 0;
    for(int i=0;i<x_size;i++)
        {
            temp = temp+x[i];
            x[i]++;
        }
    return temp;
}
long long a[]={1,2,3,4,5},n = 5;
int main()
{
    cout<<sum(a,n)<<endl;
    for(int i = 0; i<n;i++)
        cout<<a[i]<<" ";
}
Output: 15
        2 3 4 5 6
```

B. CÁC VÍ DỤ MẪU

Ví dụ 8.1: Chương trình nhập điểm toán, điểm tin, điểm tiếng Anh và in ra điểm trung bình. Trong chương trình ta thiết lập ba hàm nhap() có chức năng nhập dữ liệu, xuly() có chức năng tính điểm trung bình, xuat() có chức năng đưa dữ liệu ra màn hình.

```cpp
#include <bits/stdc++.h>
using namespace std;
float To, Ti, Anh, dtb;
string Hoten;
void nhap()
{
    cout<<"Ho ten ";          cin>>Hoten;
    cout<<"Nhap diem Toan ";cin>>To;
    cout<<"Nhap diem Tin "; cin>>Ti;
    cout<<"Nhap diem Anh "; cin>>Anh;
}
void xuly()
{
    dtb = (To+Ti+Anh)/3;
}
void xuat()
{
    cout<<"Ho ten hoc sinh"<<Hoten<<endl;
    cout<<"Toan: "<<To<<endl;
    cout<<"Tin: "<<Ti<<endl;
    cout<<"Anh: "<<Anh<<endl;
    cout<<"Diem trung binh: " << dtb;
}
int main()
{
    nhap();
    xuly();
    xuat();
}
```

```
D:\C++\FUN1.exe
Ho ten Nguyen
Nhap diem Toan 9
Nhap diem Tin 10
Nhap diem Anh 9
Ho ten hoc sinhNguyen
Toan: 9
Tin: 10
Anh: 9
Diem trung binh: 9.333
Process returned 0 (0x
Press any key to conti
```

Ví dụ 8.2: Chương trình sau đổi chỗ hai biến a, b và in ra. Chú ý ta có ký hiệu & trước các tham số hình thức a, b nhằm mục đích giữ nguyên sự thay đổi giá trị của a, b sau khi gọi hàm doicho().

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

void doicho(int &a, int &b)
{
    int x = a;
    a = b;
    b = x;
}
int main()
{
    int x = 10, y = 20;
    cout<<"Truoc khi goi ham "<<endl;
    cout<<"x = "<<x<<"y = "<<y<<endl;
    doicho(x,y);
    cout<<"Sau khi goi ham "<<endl;
    cout<<"x = "<<x<<"y = "<<y<<endl;
}
```

```
D:\C++\FUN2.exe
Truoc khi goi ham
x = 10y = 20
Sau khi goi ham
x = 20y = 10
```

Ví dụ 8.3: Chương trình sau viết hàm tìm ước chung lớn nhất của hai số và áp dụng tìm ước chung lớn nhất của n số nguyên cho trước.

```cpp
#include <bits/stdc++.h>
using namespace std;
int gcd(int a, int b)
{
    while(b!=0)
    {
        int tam = a%b;
        a = b;
        b = tam;
    }
    return a;
}
int main()
{
    int n,a[105], res;
    cin>>n;
    for(int i=1;i<=n;i++) cin>>a[i];
    res=a[1];
    for(int i=2;i<=n;i++) res=gcd(res,a[i]);
    cout<<res;
}
```

```
D:\C++\FUN3.exe
5
20 40 50 70 80
10
Process returned
Press any key to
```

Ví dụ 8.4: Chương trình sau viết hàm kiểm tra một số nguyên có là nguyên tố hay không và đếm số nguyên tố trong đoạn $[a;b]$.

```cpp
#include <bits/stdc++.h>
using namespace std;
bool NT(int x)
{
    if(x==1) return false;
    for(int i=2;i<=sqrt(x);i++)
        if(x%i==0) return false;
    return true;
}
int main()
{
    int a,b, dem=0;
    cin>>a>>b;
    for(int i=a;i<=b;i++)
        if(NT(i)) dem++;
    cout<<dem;
}
```

```
D:\C++\FUN4.exe
1 100
25
Process ret
Press any k
```

C. BÀI TẬP ÁP DỤNG

Bài 8.1 – (N0801A) Tổng các chữ số

Yêu cầu: Cho dãy số $a_1, a_2, \ldots a_n$. Hãy tính tổng các chữ số của các số hạng trong dãy số đã cho.

Ví dụ: cho dãy $(a) = (11, 22, 33, 4\,4, 55)$ có tổng các chữ số là: $1 + 1 + 2 + 2 + 3 + 3 + 4 + 4 + 5 + 5 = 30$.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n $(0 < n \leq 10^5)$,
- Dòng thứ 2 ghi n số nguyên dương kiểu 64 – bit.

Kết quả: In ra tổng các chữ số của các số trong dãy.

Ví dụ:

input	output
5 11 22 33 44 55	30

Bài 8.2 – (N0802A) Tổng chữ số chia hết cho 9

Yêu cầu: Đặt $T(x)$ là hàm tính tổng các chữ số của số nguyên x. Cho dãy số nguyên $(a_1, a_2, \dots a_n)$, hãy đếm số cặp (i, j) thỏa mãn: $1 \le i < j \le n$ và $T(a_i) + T(a_j)$ chia hết cho 9.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n $(0 < n \le 100)$,
- Dòng thứ 2 ghi n số nguyên dương kiểu 64 – bit.

Kết quả: In ra số lượng cặp (i, j) thỏa mãn yêu cầu.

Ví dụ:

input	output
3 12 30 24	2

Bài 8.3 – (N0803A) Đếm ước chung lớn nhất

Yêu cầu: Cho dãy số nguyên $(a_1, a_2, \dots a_n)$, hãy đếm số cặp (i, j) thỏa mãn: $1 \le i < j \le n$ và $gcd(a_i, a_j) > 1$ với gcd là phép lấy ước chung lớn nhất của hai số.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n $(0 < n \le 100)$,
- Dòng thứ 2 ghi n số nguyên dương kiểu 64 – bit.

Kết quả: In ra số lượng cặp (i, j) thỏa mãn yêu cầu.

Ví dụ:

input	output
3 21 6 57	3

Bài 8.4 – (N0804A) Đếm bội chung nhỏ nhất

Yêu cầu: Cho dãy số nguyên $(a_1, a_2, \dots a_n)$, hãy đếm số cặp (i, j) thỏa mãn: $1 \le i < j \le n$ và $lcm(a_i, a_j) < K$ với lcm là phép lấy bội chung nhỏ nhất của hai số.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n $(0 < n \le 100)$,
- Dòng thứ 2 ghi n số nguyên dương kiểu 64 – bit.

Kết quả: In ra số lượng cặp (i, j) thỏa mãn yêu cầu.

Ví dụ:

input	output
3 6 3 2 4	2

Bài 8.5 – (N0805A) Số ước chia hết cho 7

Yêu cầu: Đặt $P(x)$ là hàm tính số các ước số của số nguyên x. Cho dãy số nguyên $(a_1, a_2, \dots a_n)$, hãy đếm số cặp (i, j) thỏa mãn: $1 \le i < j \le n$ và $P(a_i) + P(a_j)$chia hết cho 7.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n $(0 < n \le 100)$,
- Dòng thứ 2 ghi n số nguyên dương kiểu 64 – bit.

Kết quả: In ra màn hình xâu kí tự S.

Ví dụ:

input	output
3 3 5 16	2

Bài 8.6 – (N0806A) Bội chung tổng chữ số

Yêu cầu: Viết chương trình nhập vào hai số nguyên. In ra bội chung nhỏ nhất của tổng chữ số của hai số nhập vào.

Dữ liệu: Một dòng ghi 2 số nguyên a, b kiểu 64 - bit.

Kết quả: Bội chung nhỏ nhất của tổng các chữ số của a, b.

Ví dụ:

input	output
12 34	21

Bài 8.7 – (N0807B) Nguyên tố lớn nhất

Yêu cầu: Viết chương trình tìm số nguyên tố lớn nhất không vượt quá n $(0 < n \le 10^7)$.

Dữ liệu: Một dòng ghi số nguyên dương n.

Kết quả: In ra số nguyên tố lớn nhất không vượt quá n.

Ví dụ:

input	output
10	7

Bài 8.8 – (N0808B) Nguyên tố nhỏ nhất

Yêu cầu: Viết chương trình tìm số nguyên tố nhỏ nhất lớn hơn n $(0 < n \le 10^7)$.

Dữ liệu: Một dòng ghi số nguyên dương n.

Kết quả: In ra số nguyên tố nhỏ nhất lớn hơn n.

Ví dụ:

input	output
10	11

CHƯƠNG 9 – KIỂU DỮ LIỆU STRUCT

A. KIẾN THỨC GHI NHỚ

Trong nhiều tình huống ta cần làm việc với kiểu dữ liệu có nhiều thành phần như xét một đối tượng học sinh bao gồm các thông số: họ và tên, ngày tháng năm sinh, điểm môn Toán, điểm môn Tin, …Với tình huống như vậy, ngôn ngữ lập trình C++ cung cấp kiểu dữ liệu cấu trúc – `struct`.

1. Khai báo

Cú pháp khai báo của kiểu dữ liệu `struct` như sau:

```
struct <tên kiểu dữ liệu>
    {
       <kiểu dữ liệu 1> thành phần thứ 1;
       <kiểu dữ liệu 2> thành phần thứ 2;
        …
       <kiểu dữ liệu n> thành phần thứ n;
    };
```

Sau khi khai báo như trên ta có thể sử dụng kiểu dữ liệu mới như một kiểu dữ liệu chuẩn. Chẳng hạn ta khai báo kiểu dữ liệu sau:

```
   struct HS
    {
       string hoten;
       float diemToan,diemTin;
     };
      HS hocsinh1, hocsinh2;
```

Với khai báo kiểu HS trong struct, ta sẽ sử dụng HS như một kiểu dữ liệu chuẩn.

2. Sử dụng kiểu struct

Mỗi kiểu struct thường bao gồm nhiều thành phần mà mỗi thành phần ta gọi là trường. Ví dụ kiểu HS ở trên có tới 3 trường là hoten, diemToan, diemTin.

Để truy cập vào mỗi trường ta có cú pháp sau:

```
    <Tên biến kiểu struct>.<tên bản ghi>;
```

Chẳng hạn với khai báo trên ta có thể truy cập vào trường hoten của biến hocsinh1 như sau:

```
    hocsinh1.hoten;
```

Vậy để đọc dữ liệu vào một biến struct ta đọc vào từng trường một. Tương tự như vậy để in dữ liệu một biến struct ta in ra từng trường một. Ta có thể gán dữ liệu cho từng trường của biến struct hoặc có thể gán hai biến struct cho nhau:

```
    cin>>hocsinh1.hoten;
    cin>>hocsinh1.diemToan;
    cin>>hocsinh1.diemTin;
    cout<<hocsinh1.hoten<<endl;
    cout<<hocsinh1.diemToan<<endl;
```

```
    cout<<hocsinh1.diemTin<<endl;
    hocsinh1.hoten = “Nguyen Van A”;
    hocsinh2 = hocsinh1;
```

B. CÁC VÍ DỤ MẪU

Ví dụ 9.1: Chương trình nhập vào hai phân số $\frac{a}{b}, \frac{x}{y}$ và in ra phân số tổng $\frac{a}{b} + \frac{x}{y}$ dưới dạng tối giản.

```
#include <bits/stdc++.h>
using namespace std;
struct ps
{
    int tu;//Thành phần thứ nhất là tử số
    int mau;//Thành phần thứ hai là mẫu số
};
ps cong(ps p, ps q)//Hàm tính tổng hai số
{
    ps tam;
    tam.tu = p.tu*q.mau+p.mau*q.tu;
    tam.mau = tam.tu*tam.mau;
    int gcd = __gcd(tam.tu,tam.mau);
    //Tối giản phân số
    tam.tu = tam.tu/gcd;
    tam.mau = tam.mau/gcd;
}
int main()
{
    ps a,b,c;
    cin>>a.tu>>a.mau;
    cin>>b.tu>>b.mau;
    c = cong(a,b);
    cout<<c.tu<<" "<<c.mau;
}
```

Ở đây ta đã sử dụng cấu trúc struct với hai thành phần tử và mẫu để biểu diễn cho kiểu phân số hữu tỷ.

Ví dụ 9.2: Chương trình nhập vào hai điểm và in ra khoảng cách giữa hai điểm:

```
#include <bits/stdc++.h>
using namespace std;
struct toado //Khai báo kiểu struct tọa độ
{
    int x;//Thành phần thứ nhất là hoành độ
    int y;//Thành phần thứ hai là tung độ
};
//Hàm tính khoảng cách giữa hai điểm A,B
double kc(toado A, toado B)
```

```
{
    double t1 = (A.x-B.x)*(A.x-B.x);
    double t2 = (A.y-B.y)*(A.y-B.y);
    return(sqrt(t1+t2));
}
int main()
{
    toado A, B;
    cin>>A.x>>A.y;
    cin>>B.x>>B.y;
    cout<<kc(A,B);
}
```

Ở đây ta sử dụng cấu trúc struct với hai thành phần x và y mô tả một điểm trong hệ trục tọa độ Oxy.

Ví dụ 9.3: Chương trình sau sử dụng kiểu struct để nhập vào ba điểm A, B, C và in ra diện tích tam giác ABC.

```
#include <bits/stdc++.h>
using namespace std;
struct toado //Khai báo kiểu struct tọa độ
{
    int x;//Thành phần thứ nhất là hoành độ
    int y;//Thành phần thứ hai là tung độ
};
//Hàm tính diện tích tam giác ABC
double dientich(toado A, toado B,toado C)
{
    toado AB, AC
    AB.x = B.x - A.x;
    AB.y = B.y - A.y;
    AC.x = C.x - A.x;
    AC.y = C.y - A.y;
    //Công thức tính diện tích tam giác ABC
    return(abs(AB.x*AC.y-AC.x*AB.y)/2.0);
}
int main()
{
    toado A, B, C;
    cin>>A.x>>A.y;
    cin>>B.x>>B.y;
```

```
        cin>>C.x>>C.y;
        cout<<dientich(A,B,C);
    }
```

Ở đây ta vẫn sử dụng kiểu struct để mô ta điểm và vector trong hệ trục Oxy. Công thức diện tích tam giác ở đây là: $S = \frac{1}{2}\left|x_{\overrightarrow{AB}} * y_{\overrightarrow{AC}} - x_{\overrightarrow{AC}} * y_{\overrightarrow{AB}}\right|$.

C. BÀI TẬP ÁP DỤNG

Bài 9.1 – (N0901A) Hình bình hành

Yêu cầu: Cho 3 điểm A, B, C. In ra điểm D sao cho ABCD là hình bình hành.

Dữ liệu: Một dòng ghi 6 số nguyên xa, ya, xb, yb, xc, yc là tọa độ của A, B, C.

Kết quả: Ghi ra hai số nguyên là tọa độ điểm D.

Ví dụ:

input	output
1 0 0 0 0 1	1 1

Bài 9.2 – (N0902A) Diện tích hình bình hành

Yêu cầu: Cho 3 điểm A, B, C. Tính diện tích của hình bình hành ABCD

Dữ liệu: Một dòng ghi 6 số nguyên xa, ya, xb, yb, xc, yc là tọa độ của A, B, C.

Kết quả: Ghi ra một số nguyên là diện tích của hình bình hành ABCD.

Ví dụ:

input	output
0 0 1 0 0 1	1

Bài 9.3 – (N0903B) Diện tích đa giác lồi

Yêu cầu: Cho đa giác lồi $A_1A_2\ldots A_n$. Hãy tính diện tích của đa giác lồi đó theo tọa độ của các đỉnh.

Dữ liệu gồm 2 dòng:

- Dòng 1 ghi số nguyên dương n $(0 < n \leq 10^6)$.
- n dòng kế tiếp mỗi dòng ghi 2 số nguyên x, y kiểu 32 – bit.

Kết quả

- Hai dòng ghi ra phân số tối giản biểu diễn diện tích đa giác; biết rằng diện tích của đa giác có tọa độ nguyên luôn là một phân số hữu tỷ.

Ví dụ:

input	output
3 0 0 1 0 0 1	1 2

Bài 9.4 – (N0904C) Danh sách học sinh

Yêu cầu: Có n học sinh, mỗi học sinh có 3 thông tin: Họ tên, điểm toán, điểm tin. Sắp xếp danh sách học sinh theo tiêu chí:

- Họ tên tăng dần.
- Nếu họ tên giống nhau thì tổng điểm (= điểm toán + điểm tin) tăng dần.
- Nếu họ tên và tổng điểm giống nhau thì điểm toán tăng dần.

Dữ liệu:

- Dòng đầu ghi số nguyên dương n ($n \leq 10^5$).
- n dòng sau, mỗi dòng ghi 3 thông tin: Họ tên, điểm toán, điểm tin (họ tên là dãy kí tự latinh thường, độ dài không quá 5; điểm toán và điểm tin là một số nguyên từ 0 đến 10).

Kết quả: ghi ra n dòng ghi thông tin của n học sinh được sắp xếp.

Ví dụ:

input	output
3 6 3 2 4	2

Bài 9.5 – (N0905B) Cầu thủ trẻ nhất

Yêu cầu: Thông tin của một cầu thủ bao gồm: Họ tên, cân nặng, chiều cao, tuổi. Hãy nhập vào danh sách một nhóm cầu thủ, in ra cầu thủ có tuổi trẻ nhất biết các cầu thủ có độ tuổi khác nhau.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm n ($0 < n \leq 100$)
- n dòng kế tiếp, mỗi dòng ghi thông tin của cầu thủ bao gồm theo thứ tự họ tên, cân nặng, chiều cao, tuổi.

Kết quả: In ra thông tin cầu thủ có tuổi trẻ nhất.

Ví dụ:

input	output
3 An 70 180 20 Binh 75 183 26 Cuong 70 180 30	An 70 180 20

Bài 9.6 – (N0906B) Cầu thủ trẻ nhất

Yêu cầu: Viết chương trình nhập vào hai phân số hữu tỷ, tính tổng hai phân số và in ra tổng hai phân số dưới dạng tối giản

Dữ liệu:

- Dòng 1 ghi 2 số nguyên a, b kiểu 32 bit biểu diễn phân số a/b,
- Dòng 2 ghi 2 số nguyên c, d kiểu 32 bit biểu diễn phân số c/d.

Kết quả: Ghi ra phân số tổng $a/b + c/d$ dưới dạng phân số tối giản.

Ví dụ:

input	output
3 An 70 180 20 Binh 75 183 26 Cuong 70 180 30	An 70 180 20

CHƯƠNG 10 – MỘT SỐ THUẬT TOÁN SỐ HỌC CƠ BẢN

A. KIẾN THỨC GHI NHỚ

1. Thuật toán tìm UCLN, BCNN

Thuật toán tìm ước chung lớn nhất là một thuật toán cơ bản được đề cập đến trong SGK Tin học 10 hiện hành. Giải thuật này được Euclide đưa ra trong cuốn *Element* vào năm 300 trước công nguyên. Thuật toán này dựa trên bổ đề sau đây:

"Với a, b là hai số nguyên $(a \geq b)$, ta thực hiện chia a cho b được thương q, số dư r $(r \geq 0)$ tức là $a = bq + r$, khi đó ta có: $\text{UCLN}(a,b) = \begin{cases} b & nếu\ r = 0 \\ \text{UCLN}(b,r) & nếu\ r \neq 0 \end{cases}$"

Từ đó ta có thuật toán tìm UCLN của hai số nguyên dương a, b như sau:

```
long long UCLN(long long a, long long b) {
     long long temp;
      while (b != 0)
      {
          temp = a % b;
          a = b;
          b = temp;
      }
      return a;
 }
```

Giải thuật trên có độ phức tạp $O(log(max(a,b))$. Ta chú ý rằng có một phiên bản khác của thuật toán tìm UCLN như sau:

```
long long UCLN(long long a, long long b)
{
     long long temp;
      while (a!= b)
      {
          if(a>b) a = a - b;
          if(b>a) b = b - a;
      }
      return a;
 }
```

Thuật toán này trong trường hợp xấu nhất có thể độ phức tạp của nó là $O(max(a,b))$ và sẽ thời gian chạy sẽ vượt quá mức $1s$ với a, b có khoảng cách cỡ 10^9.

2. Thuật toán kiểm tra số nguyên tố

- Thuật toán duyệt hết ước:

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

```
bool Check(long long x)
{
    if(x==1) return false;
    for(long long i=2;i<x;i++)
        if(x%i==0) return false;
    return true;
}
int main()
{
    long long n;
    cin>>n;
    if(Check(n)) cout<<"YES";
    else cout<<"NO";
}
```

- Thuật toán duyệt ước đến $\sqrt{n}$:

✓ Nhận xét thấy nếu n có một ước nguyên dương i thì nó sẽ có ước nguyên dương n/i.

✓ Từ đó ta chỉ cần kiểm tra xem nếu n không chia hết cho các phần tử từ 2 đến $\sqrt{n}$ thì n là số nguyên tố.

Từ đó ta có hàm kiểm tra nguyên tố như sau:

```
#include <bits/stdc++.h>
using namespace std;
bool Check(long long x)
{
    for(long long i=2;i*i<=x;i++)
        if(x%i==0) return false;
    return x>1;
}
int main()
{
    long long n;
    cin>>n;
    if(Check(n)) cout<<"YES";
    else cout<<"NO";
}
```

3. Giải thuật sàng nguyên tố

Khi liệt kê các số nguyên tố nhỏ hơn n ta có thể sử dụng hàm kiểm tra nguyên tố ở trong mục 3 như sau:

```
for(long long i = 1; i  <= n; ++i)
if(Check(i)) cout<<i<<endl;
```

Khi đó với n cỡ 10^7 việc sử dụng thuật toán ở trên sẽ vượt quá tốc độ xử lý của máy tính hiện nay vì tốc độ máy tính hiện tại chỉ có thể thực hiện tối đa 10^8 phép toán trên một giây mà trong trường hợp trên đã có cỡ 10^9 lệnh thực thi. Vì vậy trong trường hợp này ta nên sử dụng thuật toán sàng nguyên tố nhằm lưu toàn bộ số nguyên tố vào một mảng `isPrime`[] như sau:

- Đánh dấu tất cả mọi số từ 1 đến n là true;
- Hiển nhiên số 1 không là số nguyên tố, ta gán `isPrime`[1] = **`false`**;
- Duyệt từ 2 đến $\sqrt{n}$, lần lượt đánh dấu các bội của các số nguyên tố, đầu tiên 2 là số nguyên tố, ta đánh dấu các bội của 2 là hợp số, sau đó lần lượt các số kế tiếp ta sẽ đánh dấu các bội của nó bằng đoạn mã sau:

```
for(long long i = 2; i * i <= nmax; ++i)
if(isPrime[i])
        for(long long j = i * i; j <= nmax; j += i)
                isPrime[j] = false;
```

Khi đó mảng isPrime[] sẽ đánh dấu tính nguyên tố của các số nguyên.

4. Tính chất đồng dư

Ta cần đặc biệt lưu ý đến tính chất đồng dư sau đây:

- $\big((a\%m) + (b\%m)\big)\%m = (a + b)\%m$
- $\big((a\%m) * (b\%m)\big)\%m = (a * b)\%m$

Trong một số bài toán vì giới hạn biểu diễn của kiểu nguyên trong ngôn ngữ lập trình C^{++} nên thường có yêu cầu chia lấy dư cho một số nguyên khi in ra kết quả. Nếu tính toán đến bước cuối cùng mới chia lấy dư thì kết quả sẽ bị sai lệch do tràn số vượt khả năng biểu diễn của kiểu dữ liệu chuẩn nên ta sẽ chia lấy dư trong quá trình tính toán.

Ví dụ cần tính $(n!)\%m$ ta thường sử dụng vòng lặp như sau:

```
for(long long i = 1; i <= n; ++i)
            gt = (gt*i)%m;
```

Đối với phép chia lại không có được tính chất phân phối đồng dư như nhân và cộng. Nghĩa là ta không có tính chất dạng $\frac{a\%m}{b\%m} = \left(\frac{a}{b}\right)\%\, m$. Trong trường hợp này ta sẽ có một số phương pháp tìm nghịch đảo modulo như vận dụng định lý Fermat bé như sau:

Định lý Fermat: Nếu p là một số nguyên tố và b là số nguyên dương bất kỳ thì ta có $b^p - b$ chia hết cho p.

Ta gọi nghịch đảo modulo của b theo modulo m là số nguyên c sao cho $b.c \equiv 1(mod\ m)$ hay là $b.c$ chia m dư 1. Nếu ta tìm được số nguyên c như vậy thì ta có tính chất sau: $\left(\frac{a}{b}\right) \equiv \left(\frac{a.c}{b.c}\right) \equiv ac(mod\ m)$.

Từ định lý Fermat nhỏ ta có cách tính $\left(\frac{a}{b}\right)\%\,p$ với p là số nguyên tố như sau:

- Ta có phép biến đổi: $b^p - b \vdots p \Leftrightarrow b(b^{p-1}-1) \vdots p \Leftrightarrow b^{p-1}-1 \vdots p$ hay $b^{p-1} \equiv 1 (mod\ p)$.
- Suy ra $\left(\frac{a}{b}\right)\%p = \left[\left(\frac{a}{b}\right) \times b^{p-1}\%\right]p = (a \times b^{p-2})\%p$.
- Xây dựng hàm tính b^{p-2} bằng chia để trị (xem chương 11).
- Ta có công thức: $\left(\frac{a}{b}\right)\%p = (a \times b^{p-2})\%p$ và đưa về nhân đồng dư.

```
ll pw(ll a,ll n, ll mod)
{
    if(n==0) return 1;
    ll t=pw(a,n/2, mod)%mod;
    t=t*t%mod;
    if(n%2==1) t=(t*a)%mod;
    return t;
}
int main()
{
    ll a,b,p;
    cin>>a>>b>>p;
    cout<< (a * pw(b,p-2, p))%p;
}
```

B. CÁC VÍ DỤ MẪU

Ví dụ 10.1: Chương trình đếm số lượng số nguyên tố trong đoạn [a;b].

```
#include <bits/stdc++.h>
using namespace std;
bool Check(long long x)
{
    if(x==1) return false;
    for(long long i=2;i*i<=x;i++)
        if(x%i==0) return false;
    return true;
}
int main()
{
    long long a,b,dem = 0;
    cin>>a>>b;
    for(long long i=a;i<=bi++)
        if(Check(i)) dem++;
```

```
    cout<<dem;
}
```

Ví dụ 10.2: Chương trình sau sử dụng sàng nguyên tố để liệt kê số nguyên tố trong đoạn $[a; b]$.

```
#include <bits/stdc++.h>
#define nmax 10000007
using namespace std;
bool isPrime[nmax];
void sieve()
{
    memset(isPrime,true,sizeof(isPrime));
    isPrime[1] = false;
    for(long long i = 2; i * i <= nmax; ++i)
        if(isPrime[i]
           for(long long j = i * i; j <= nmax; j += i)
               isPrime[j] = false;
}
int main()
{
    sieve();
    int n;
    cin>>n;
    for(int i=1;i<=n;i++)
        if(isPrime[i]) cout<<i<<'\n';
}
```

Ví dụ 10.3: Chương trình sau phân tích đếm số lượng ước nguyên dương của số nguyên dương n.

Tương tự **Ví dụ 10.1**, nếu n có ước nguyên dương là i thì n cũng có ước tương ứng là n/i. Vì vậy ta chỉ cần duyệt đến $\sqrt{n}$, mỗi lần n chia hết cho i ta sẽ cập nhật luôn 2 ước. Với cách làm này ta có thể đếm số ước của n lên đến10^{12} trong thời gian $1s$. Chú ý trong trường hợp n là số chính phương thì ta cần trừ đi 1 ước do $\sqrt{n} = n/\sqrt{n}$.

```
#include <bits/stdc++.h>
using namespace std;
long long n, i, dem = 0;
int main()
{
    cin>>n;
```

```
    for(i=1;i*i<=n;i++)
        if(n%i==0) dem+=2;
    if(n==i*i) dem--;
    cout<<dem;
}
```

C. BÀI TẬP ÁP DỤNG

Bài 10.1 – (N1001A) Số lượng số nguyên tố

Yêu cầu: Cho dãy số $a_1, a_2, \dots a_n$. Hãy đếm số nguyên tố có mặt trong dãy đã cho.

Dữ liệu:

- Dòng đầu tiên ghi số nguyên không âm $n(\ 0 < n \leq 10^3)$,
- Dòng thứ 2 ghi n số nguyên dương kiểu 32 – bit.

Kết quả: in ra số lượng số nguyên tố có mặt trong dãy.

Ví dụ:

input	output
5 2 5 7 9 11	4

Bài 10.2 – (N1002B) Số nguyên tố trong đoạn

Yêu cầu: Đếm số lượng số nguyên tố trong đoạn [a; b].

Dữ liệu: Một dòng ghi hai số nguyên a, b với $0 < a < b \leq 10^7$.

Kết quả: Một dòng là số lượng số nguyên tố trong đoạn $[a;\ b]$.

Ví dụ:

input	output
1 10	4

Bài 10.3 – (N1003B) Số đặc biệt

Yêu cầu: Ta gọi dãy $a_1, a_2, \dots a_n$ là dãy đặc biệt nếu tồn tại số nguyên dương $x > 1$ sao cho tất cả các phần tử của dãy trên đều chia hết cho x. Cho một dãy đặc biệt A và số nguyên dương k. Hãy tìm số nguyên l lớn nhất thỏa mãn: $0 \leq l \leq k$ và $A \cup \{l\}$ là dãy đặc biệt.

Dữ liệu:

- Dòng đầu tiên chứa hai số nguyên n và k,
- Dòng thứ 2 chứa dãy đặc biệt A.

Kết quả: Một dòng là số nguyên dương k thỏa mãn bài toán.

Ví dụ:

input	output
3 5	4

2 6 4	

Bài 10.4 – (N1004C) Số bin bon

Yêu cầu: Mr Bon là cậu em trai song sinh của Mr Bin. Cậu ấy cũng đang học các con số, cậu ấy tính tổng các chữ số của một số nguyên dương n cho trước được một số nguyên dương a. Tiếp đến cậu lại đi tìm chữ số lớn nhất của n và viết ghép chữ số lớn nhất đó vào sau số a thu được số nguyên dương k. Mr Bon muốn biết k có phải là số nguyên tố hay không? Nếu số đó là nguyên tố thì cậu gọi số đó là số "binbon". Ví dụ số 111 là một số "binbon" vì tổng các chữ số là $a = 3$, chữ số lớn nhất là 1 nên số k lập được là 31, số 31 là một số nguyên tố nên $n = 111$ là số "binbon". Thú vị với suy nghĩ này cậu bé muốn kiểm tra một loạt các số có phải là số "binbon" hay không? Hãy giúp cậu bé nhé!

Dữ liệu:

- Dòng đầu ghi số nguyên dương t là số test ($0 < t \leq 1000$).
- t dòng sau mỗi dòng ghi số nguyên dương n ($0 < n \leq 10^{1000}$).

Kết quả: in ra t dòng, mỗi dòng ghi YES hoặc NO nếu số tương ứng là "binbon" hoặc không?

Ví dụ:

input	output
2 123 111	NO YES

Bài 10.5 – (N1005B) Số nguyên tố fibonacci

Yêu cầu: Viết chương trình nhập vào số nguyên dương n. In ra số nguyên tố là số Fibonacci lớn nhất không vượt quá n.

Dữ liệu: Một dòng ghi 1 số nguyên dương $n (0 < n < 10^{12})$.

Kết quả: Ghi ra số nguyên tố Fibonacci lớn nhất không vượt quá n.

Ví dụ:

input	output
10	5

Bài 10.6 – (N1006A) Cơ số k

Yêu cầu: Một số nguyên khi biểu diễn trong hệ cơ số k chỉ có đúng k ký tự để biểu diễn. Ví dụ trong hệ nhị phân chỉ có 2 ký tự 0 và 1. Số 8 trong hệ 10 được biểu diễn trong hệ nhị phân như sau:100(2), số 15 là dãy bít 1111(2). Cho một số nguyên dương n và một số nguyên dương k. Hãy đưa ra cách biểu diễn n trong hệ cơ số k.

Input: Một dòng duy nhất chứa hai số nguyên dương $n, k (n \leq 10^9, k \leq 9)$.

Output: In ra số n trong hệ cơ số k.

Ví dụ:

input	output
15 2	1111

Bài 10.7 – (N1007B) Ước

Yêu cầu: Hãy đếm được có bao nhiêu số có đúng 3 ước dương trong n số nguyên không âm.

Dữ liệu:

- Dòng 1 chứa số nguyên dương $n\left(0<n\le 10^5\right)$,
- Dòng 2 chứa n số nguyên dương $a_i\left(0<a_i\le 10^{12}\right)$.

Kết quả: In ra n dòng, dòng thứ i ghi YES nếu a_i có đúng 3 ước dương, còn ngược lại thì ghi NO.

Ví dụ:

input	output
3 4 10 12	YES NO NO

Bài 10.8 – (N1008B) Số thừa số nguyên tố

Yêu cầu: Hãy đếm số thừa số nguyên tố có trong số tự nhiên.

Input: Một dòng ghi số nguyên $n\ \left(0<n\le 10^{12}\right)$.

Output: ghi ra số lượng thừa số nguyên tố có trong n.

Ví dụ:

input	output
10	2

Bài 10.9 – (N1009C) Số siêu nguyên tố

Yêu cầu: Ta định nghĩa n là số siêu nguyên tố nếu n là một số nguyên tố và khi liên tục bỏ đi một chữ số bên phải n vẫn là số nguyên tố cho tới khi n chỉ còn 1 chữ số. Ví dụ, số 23 là số siêu nguyên tố vì 23 và 2 là các số nguyên tố, ngoài ra 413 không phải là số siêu nguyên tố vì 3 nguyên tố, 41 nguyên tố nhưng 4 không nguyên tố. Viết chương trình nhập vào số nguyên dương n. In ra YES nếu n là siêu nguyên tố, in ra NO nếu n không là siêu nguyên tố.

Dữ liệu vào: Dòng đầu tiên ghi số nguyên dương $n\left(0<n\le 10^9\right)$.

Dữ liệu ra: In ra YES nếu n là số siêu nguyên tố, NO nếu n ngược lại.

Ví dụ:

input	output
23	YES

Bài 10.10 – (N1010D) Số supper nguyên tố

Yêu cầu: Ta mở rộng thêm khái niệm số nguyên tố theo hướng sau. Ta liệt kê các số nguyên tố từ 2 trở đi: $2, 3, 5, 7, 11, 13, \ldots$Một số ta gọi là supper nguyên tố nếu chỉ số của nó cũng là nguyên tố. Ví dụ 3 là một số supper nguyên tố do 3 là số nguyên tố thứ 2 và 2 là một số nguyên tố, ngoài ra 5, 11 cũng là các số supper nguyên tố.

Yêu cầu: Viết chương trình in ra số supper nguyên tố thứ n.

Input: Một dòng chứa số nguyên dương$n (0 < n \leq 500)$.

Output: Số supper nguyên tố thứ n.

Ví dụ:

input	output
3	11

Bài 10.11 – (N1011E) Định đề Bertrand

Yêu cầu: Định đề Bertrand phát biểu rằng với bất kỳ số nguyên n, luôn tồn tại ít nhất một số nguyên tố P sao cho $n + 1 \leq p \leq 2n$. Cho n, đếm số lượng số nguyên tố thuộc đoạn $[n + 1; 2n]$.

Input:

- Dòng 1 ghi số T là số bộ dữ liệu $(0 < T \leq 10^6)$,
- T dòng tiếp theo, mỗi dòng ghi số nguyên n tương ứng $(0 < n \leq 10^6)$.

Output: gồm T dòng, mỗi dòng ghi số lượng số nguyên tố tương ứng với dữ liệu vào.

Ví dụ:

input	output
10	11

Bài 10.12 – (N1012E) Liệt kê số siêu nguyên tố

Yêu cầu: Với khái niệm số siêu nguyên tố trong bài 10.9 hãy viết chương trình đếm số lượng số siêu nguyên tố có số chữ số là n.

Dữ liệu vào: Dòng đầu tiên ghi số nguyên dương $(0 < n \leq 10)$.

Dữ liệu ra: In ra số lượng số siêu nguyên tố có số chữ số là n.

Ví dụ:

input	output
3	11

Bài 10.13 – (N1013E) Tổng phần nguyên

Yêu cầu: Cho số nguyên dương n. Hãy tính $S = \left[\frac{n}{1}\right] + \left[\frac{n}{2}\right] + \cdots + \left[\frac{n}{n}\right]$.

Dữ liệu vào: Dòng đầu tiên ghi số nguyên dương n ($0 < n \leq 10^9$).

Dữ liệu ra: In ra tổng$S$.

Ví dụ:

input	output
5	10

CHƯƠNG 11 – ĐỆ QUY

A. KIẾN THỨC GHI NHỚ

1. Khái niệm đệ quy

Chương trình con đệ quy là các chương trình con có lời gọi nó trong chính chương trình của nó. Điều này có thể hơi khác lạ với phương thức lập trình logic tuần tự nhưng đây lại là một điểm mạnh của các ngôn ngữ lập trình với việc quản lý ngăn xếp chương trình con để cất ngữ cảnh hiện tại và tiếp tục giải bài toán ở mức nhỏ hơn cho đến khi gặp trường hợp nhỏ nhất. Chương trình con đệ quy có ưu thế khi giải quyết các bài toán có dạng công thức truy hồi, các bài toán về chia để trị, các bài toán quay lui vét cạn. Một chương trình đệ quy thường có cấu trúc như sau:

```
<kiểu dữ liệu>  <tên hàm> (danh sách tham số)
{
   if(trường hợp dừng đệ quy)
       return <kết quả dừng>;
   else
        return <tên hàm>(tham số cấp nhỏ hơn);
}
```

Ta chú ý rằng chương trình con đệ quy phải có hai phần rõ rệt. Một là phần trường hợp dừng – trường hợp nhỏ nhất của bài toán, không có phần trường hợp dừng này hoàn toàn chương trình sẽ lặp tới mức tràn ngăn xếp chương trình con. Hai là trường hợp gọi đệ quy, trường hợp này sẽ đưa bài toán đang giải về bài toán cùng dạng với cấp độ nhỏ hơn. Khi có lời gọi đệ quy, trình biên dịch sẽ liên tục cất ngữ cảnh vào bộ nhớ ngăn xếp chương trình con cho tới trường hợp dừng sẽ lại quay trở lại lấy giá trị các biến trong ngăn xếp chương trình con để tính ngược lên lời gọi đầu tiên. Vì vậy khi gọi đệ quy chúng ta lưu ý không để lời gọi quá sâu để dẫn tới việc tràn bộ nhớ ngăn xếp chương trình con.

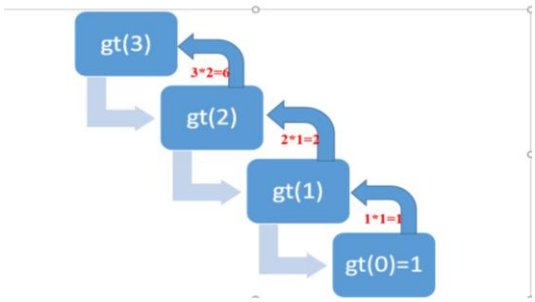
2. Ví dụ minh họa

Ta xét một ví dụ mở đầu về đệ quy như sau: Tính $n!$

```cpp
#include <bits/stdc++.h>
using namespace std;
long long gt(long long n){
    if(n==0) return 1;// Trường hợp dừng
    else return n*gt(n-1);//Lời gọi đệ quy cấp thấp hơn
}
int main()
{
    long long n;
    cin>>n;
    cout<<gt(n);
```

```
}
```

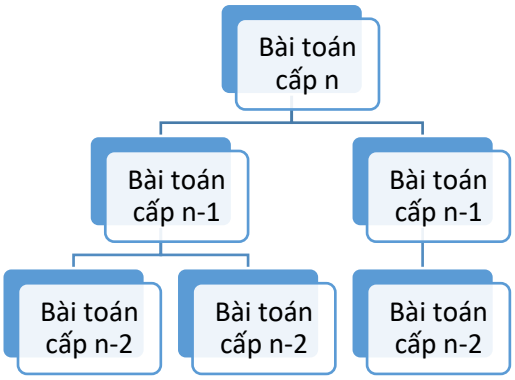
Với chương trình trên khi nhập n bằng 3 ta có một sơ đồ đệ quy như sau:



Chú ý rằng trong chương trình con đệ quy dứt khoát phải có trường hợp dừng, khi đó mới thoát ra khỏi vòng lặp đệ quy. Câu lệnh đệ quy thường ngắn gọn, súc tích, gần với bản chất toán học của bài toán. Tuy nhiên việc gọi đệ quy quá sâu sẽ phải cất ngữ cảnh quá nhiều dẫn tới tràn ngăn xếp chương trình con. Mỗi trình biên dịch ngôn ngữ thường được cấp một dung lượng nhất định cho ngăn xếp chương trình con, ta cũng có thể mở rộng trong phạm vi cho phép, tuy nhiên nó sẽ phụ thuộc nhiều vào bộ nhớ trên máy tính. Thực tế, với việc tính $n!$ theo cách viết đệ quy không đem lại ưu thế. Vì vậy đây chỉ là một ví dụ mô tả cho đệ quy mà thôi, chúng ta ít sử dụng nó trực tiếp trong tính toán như trên.

3. Một số ứng dụng của đệ quy

Đệ quy có nhiều ứng dụng trong lập trình. Một trong những ứng dụng thường gặp là chia để trị và liệt kê tổ hợp. Cấu trúc của các giải thuật chia để trị đưa việc giải quyết bài toán về giải nhiều bài toán cùng dạng nhưng có cấp thấp hơn theo mô hình cây và sử dụng được lời giải ở cấp thấp hơn để độ phức tạp bài toán giảm xuống:



Điển hình cho dạng toán này được mô tả trong **Ví dụ 11.1**

Ngoài ra để liệt kê các cấu hình tổ hợp thì cấu trúc đệ quy quay lui rất thuận tiện cho việc liệt kê này.

Ta có mô hình liệt kê tổ hợp như sau:

```
void Try (int i)//Xây dựng thành phần thứ i
{
     <Xác định Si>;
     for(xi thuộc Si)
          {
               <ghi nhận thành phần thứ i>
               if (tìm thấy nghiệm) <in ra nghiệm>;
                    else Try(i+1);
               <loại thành phần i>;
          }
}
```

Trong đó **Si** là tập hợp các giá trị mà **xi** có thể nhận được với **xi** là thành phần thứ **i** của cấu hình ta cần xây dựng.

Ta xem xét ví dụ minh họa cho dạng toán này trong **Ví dụ 11.2**

B. CÁC VÍ DỤ MẪU

Ví dụ 11.1: Xét chương trình tính $a^n\%(10^9+7)$ bằng chia để trị với độ phức tạp $O(logn)$.

```
#include <bits/stdc++.h>
#define mod 1000000007
#define ll long long
using namespace std;
ll mu(ll a, ll n) // tính a^n%mod bằng đệ quy chia để trị
{
    if(n==0) return 1;
    ll tam = mu(a,n/2);
    tam = (tam*tam)%mod;
    if(n%2==1) tam = (tam*a)%mod;
    return tam;
}
int main()
{
    long long a,n;
    cin>>a>>n;
    cout<<mu(a,n); // độ phức tạp thuật toán là O(logn)
}
```

Chú ý rằng nếu chúng ta sử dụng vòng lặp for để tính a^n thì độ phức tạp của giải thuật sẽ là $O(n)$ – chỉ có thể giải được cho n cỡ $10^6 \to 10^8$ trong thời gian 1 giây.

Ví dụ 11.2: Ta xét ví dụ liệt kê dãy nhị phân có độ dài n như sau:

```
#include<bits/stdc++.h>
#define ll long long
using namespace std;
int x[30],n;
void Xuat()
{
    for(int k=1;k<=n;k++) cout<<x[k];
    cout<<endl;
}
void Try(ll i)
{
    for(int j=0;j<=1;j++)
    {
        x[i]=j;
        if(i==n) Xuat();
            else Try(i+1);
    }
}
int main()
{
    cin>>n;
    Try(1);
}
```

Lời gọi đệ quy vét cạn thường bắt đầu với $i = 1$ nghĩa là xây dựng thành phần thứ nhất của i.

Ví dụ 11.3: Chương trình sau sẽ liệt kê ra các hoán vị của tập $X = \{1.2.\ldots, n\}$. Ta biết khi xây dựng hoán vị thì các thành phần xây dựng không được phép lặp lại. Vì vậy ta phải sử dụng một mảng đánh dấu **d[]** để đánh dấu cho các giá trị đã được chọn để không chọn lại. Đoạn chương trình liệt kê hoán vị như sau:

```
#include <bits/stdc++.h>
#define ll long long
#define fo(i,a,b) for(int i=a;i<=b;i++)
using namespace std;
ll n,x[30],d[30];
void Xuat()
{
            fo(k,1,n)  cout<<x[k];
            cout<<'\n';
}
```

```
ll Try(ll i)
{
    fo(j,1,n)
    if(d[j]==0)// nếu j chưa được chọn
    {
        x[i]=j;//chọn x[i] là j
        d[j]=1;//đánh dấu j đã được chọn
        if(i==n) Xuat();// in ra khi chọn tới xn
        else Try(i+1); // chưa chọn đến xn thì chọn tiếp
        d[j]=0;// quay lại bỏ đánh dấu để chọn tiếp sau
    }
}
int main()
{
    cin>>n;
    Try(1);
}
```

Bên cạnh phương pháp này kể từ phiên bản C14, ta có hàm `next_permutation()` để sinh hoán vị kế tiếp, đây là một cách code khá gọn gàng như sau:

```
#include<bits/stdc++.h>
using namespace std;
long long n,a[30];
int main()
{
    cin>>n;
    for(int i=1;i<=n;i++) a[i]=i;
    do
    {
        for(int i=1;i<=n;i++)  cout<<a[i];
           cout<<'\n';
    }
        while(next_permutation(a+1,a+n+1));
}
```

Bên cạnh hàm `next_permutation()` ta còn có hàm `prev_permutation()` lấy hoán vị trước đó theo thứ tự từ điển.

C. BÀI TẬP ÁP DỤNG

Bài 11.1 – (N1101A) Số tập con

Yêu cầu: Hãy đếm số tập hợp con của tập $X = \{1. 2. \dots, n\}$ biết số tập con của tập X là 2^n.

Dữ liệu: Một dòng ghi số nguyên không âm n $(0 < n \leq 10^9)$.

Kết quả: In ra số các tập con của tập X. Kết quả có thể rất lớn nên ta sẽ chia lấy dư cho $10^9 + 7$ khi in ra.

Ví dụ:

input	output
3	8

Bài 11.2 – (N1102A) Liệt kê nhị phân

Yêu cầu: Hãy in ra các dãy nhị phân độ dài n

Input: Một dòng ghi số nguyên n $(0 < n \leq 20)$.

Output: In ra các dãy nhị phân theo thứ tự từ điển.

Ví dụ:

input	output
2	00 01 10 11

Bài 11.3 – (N1103A) Liệt kê tam phân

Yêu cầu: Dãy tam phân là dãy chỉ gồm các ký tự 0, 1, 2 dùng để biển diễn các số trong hệ cơ số 3. Ví dụ ta có các dãy tam phân độ dài 2 là: 00, 01, 02, 10, 11, 12, 20, 21, 22. Hãy liệt kê theo thứ tự từ điển các dãy tam phân độ dài n.

Dữ liệu: Một dòng ghi số nguyên không âm n $(0 < n \leq 10)$.

Kết quả: In ra các dãy tam phân theo thứ tự từ điển.

Ví dụ:

input	output
2	00 01 02 10 11 12 20 21 22

Bài 11.4 – (N1104B) Liệt kê chỉnh hợp

Yêu cầu: Trong toán học, chỉnh hợp là cách chọn những phần tử từ một nhóm lớn hơn và có phân biệt thứ tự, trái với tổ hợp là không phân biệt thứ tự.

Theo định nghĩa, chỉnh hợp chập k của n phần tử là bộ sắp thứ tự gồm k phần tử của tập hợp gồm n phần tử. Hãy liệt kê các chỉnh hợp chập k của n phần tử của $X = \{1,2,\dots,n\}$

Dữ liệu nhập: Một dòng một gồm k, n $(1 \le k \le n \le 8)$.

Kết quả: Mỗi dòng in một chỉnh hợp chập k của n, các chỉnh hợp in theo thứ tự từ điển.

Ví dụ:

input	output
2 3	1 2 1 3 2 1 2 3 3 1 3 2

Bài 11.5 – (N1105C) Liệt kê chỉnh hợp tập A

Yêu cầu: Trong bài tập này bạn được cho một tập hợp A gồm n phần tử số nguyên khác nhau. Bạn hãy liệt kê các chỉnh hợp không lặp chập k của n phần tử này.

Dữ liệu nhập:

- Dòng một gồm k, n $(1 \le k \le n \le 8)$.
- Dòng hai là n phần tử của tập A.

Kết quả:

- Mỗi dòng in một chỉnh hợp chập k củan, các chỉnh hợp được in theo thứ tự từ điển.
- Dòng cuối cùng in số lượng chỉnh hợp liệt kê được.

Ví dụ:

input	output
2 3 1 9 4	1 4 1 9 4 1 4 9 9 1 9 4 6

Bài 11.6 – (N1106B) Liệt kê tổ hợp

Yêu cầu: Trong toán học, tổ hợp chập k của n là cách chọn tập con gồm k phần tử trong tập có n phần tử. Ở đây, bạn được yêu cầu in ra tất cả các tổ hợp chập k của tập $X = \{1,2,\dots,n\}$.

Dữ liệu: Một dòng một gồm k, n $(1 \le k \le n \le 8)$.

Kết quả:

- Mỗi dòng in một tổ hợp chập kcủa theo thứ tự từ điển.
- Dòng cuối cùng in số lượng tổ hợp liệt kê được.

Ví dụ:

input	output
2 3	1 2 1 3 2 3 3

Bài 11.7 – (N1107C) Liệt kê tổ hợp tập A

Yêu cầu: Hãy in ra các tổ hợp chập k của tập số nguyên A gồm n phần tử.

Dữ liệu:

- Dòng đầu tiên một gồm hai số nguyên k, n $(1 \leq k \leq n \leq 8)$.
- Dòng tiếp theo gồm n số nguyên trong tập A.

Kết quả:

- Mỗi dòng in một tổ hợp chập k của theo thứ tự từ điển.
- Dòng cuối cùng in số lượng tổ hợp liệt kê được.

Ví dụ:

input	output
2 3 1 9 4	1 4 1 9 4 9 3

Bài 11.8 – (N1108B) Liệt kê xâu ký tự AB

Yêu cầu: Hãy liệt kê các xâu độ dài n chỉ gồm hai kí tự 'A' hoặc 'B' mà không có hai kí tự 'B' nào đứng cạnh nhau.

Dữ liệu: Một dòng ghi số nguyên n $(n \leq 20)$.

Kết quả: In ra các xâu ký tự theo thứ tự từ điển.

Ví dụ:

input	output
3	AAA AAB ABA BAA BAB

Bài 11.9 – (N1109D) Liệt kê xâu hợp lệ

Yêu cầu: Hãy liệt kê các cách đặt n dấu mở ngoặc và n dấu đóng ngoặc sao cho biểu thức đó hợp lệ.

Dữ liệu: Một dòng ghi số nguyên n ($n \leq 20$).

Kết quả: In ra các xâu ký tự theo thứ tự từ điển.

Ví dụ:

input	output
3	AAA AAB ABA BAA BAB

Bài 11.10 – (N1110E) Liệt kê xâu con

Yêu cầu: Hãy liệt kê tất cả các xâu con khác nhau của xâu S.

Dữ liệu: Một dòng ghi xâu S $(S.length \leq 15)$ gồm các ký tự từ a đến z.

Kết quả: In ra các xâu con khác nhau của S theo thứ tự từ điển.

Ví dụ:

input	output
abc	a ab abc ac b bc c

Bài 11.11– (N1111D) Số mũ 1

Yêu cầu: Cho hai số nguyên a, n. Tính $a^n \% (10^9 + 7)$.

Dữ liệu: Một dòng ghi hai số nguyên a, n.

Kết quả: In ra kết quả là tổng $a^n \% (10^9 + 7)$.

Ví dụ:

input	output
2 3	8

Bài 11.12 – (N1112E) Số mũ 2

Yêu cầu: Cho số nguyên n, K tính $S = (1.2^0 + 2.2^1 + \cdots + n2^{n-1}) \% K$.

Dữ liệu: Một dòng ghi hai số nguyên n, K $(0 < n, K \leq 10^9)$.

Kết quả: In ra kết quả là tổng S.

Ví dụ:

input	output
1 10	1

Bài 11.13 – (N1113E) Số mũ 3

Yêu cầu: Cho số nguyên n, tính $S = 1 + a^1 + a^2 + \cdots + a^n$. Kết quả có thể rất lớn nên chia lấy dư cho $10^9 + 7$ khi in ra.

Dữ liệu: Một dòng ghi hai số nguyên a, n $(0 < a, n \leq 10^9)$.

Kết quả: In ra kết quả là tổng $S\%(10^9 + 7)$.

Ví dụ:

input	output
2 9	1023

Bài 11.14 – (N1114E) Quân hậu

Yêu cầu: Cho bàn cờ vua có kích cỡ $n \times n$. Hãy tìm cách xếp n quân hậu trên bàn cờ sao cho các quân hậu không ăn được nhau.

Dữ liệu: Một dòng ghi một số nguyên dương n $(0 < n \leq 12)$.

Kết quả: In ra kết quả là số cách sắp xếp n quân hậu trên bàn cờ.

Ví dụ:

input	output
8	92

LỜI KẾT

Cuốn tài liệu mong muốn cung cấp cho người học một lộ trình học tập nhập môn lập trình thuật toán với C++ từ những dòng code đơn giản nhất. Cuốn sách chia làm 12 chương với các kiến thức cơ bản về nhập môn lập trình thuật toán. Sau cuốn này bạn có thể đọc cuốn tiếp theo của bộ sách – quyển Trung sẽ cung cấp các kiến thức nền tảng để bạn có thể nghiên cứu sâu hơn về giải thuật. Trong quyển Trung sẽ tập hợp các chuyên đề về các kiểu dữ liệu nâng cao : vector, stack, queue, dequeue, các chuyên đề về đồ thị, các chuyên đề về quy hoạch động, các thuật toán xử lý xâu, các thuật toán nâng cao số học.

Hy vọng bạn đọc sẽ có những góp ý thẳng thắn để tài liệu mỗi ngày được hoàn thiện hơn.

Thành Vinh, một ngày đẹp trời tháng 04 năm 2020.