Mục Lục

[**BÀI 7: CHUẨN HÓA 1** 2](#_Toc132187419)

[**BÀI 5: PHÉP CỘNG** 4](#_Toc132187420)

[**BÀI 6: SỐ TĂNG GIẢM** 6](#_Toc132187421)

[**BÀI 8: CHUẨN HÓA 2** 9](#_Toc132187422)

[**BÀI 10: CHIA HẾT CHO 2** 11](#_Toc132187423)

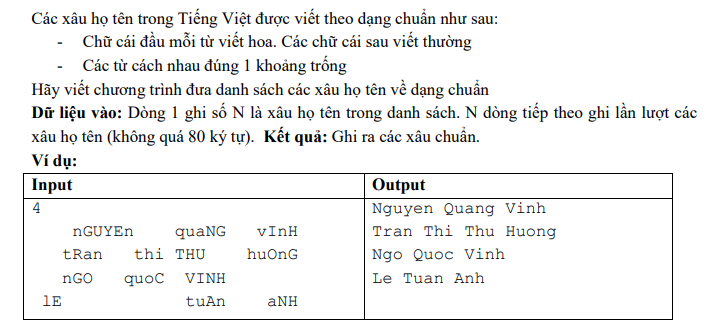
[**BÀI 1: ƯỚC SỐ CHUNG LỚN NHẤT VÀ BỘI CHUNG NHỎ NHẤT** 13](#_Toc132187424)

[**BÀI 2: BẮT ĐẦU VÀ KẾT THỨC** 15](#_Toc132187425)

[**BÀI 3: MẢNG ĐỐI XỨNG** 18](#_Toc132187426)

[**BÀI 4: PHÂN TÍNH THỪA SỐ NGUYÊN TỐ.** 20](#_Toc132187427)

# **BÀI 7: CHUẨN HÓA 1**

****

1. Xác định Input/Output.

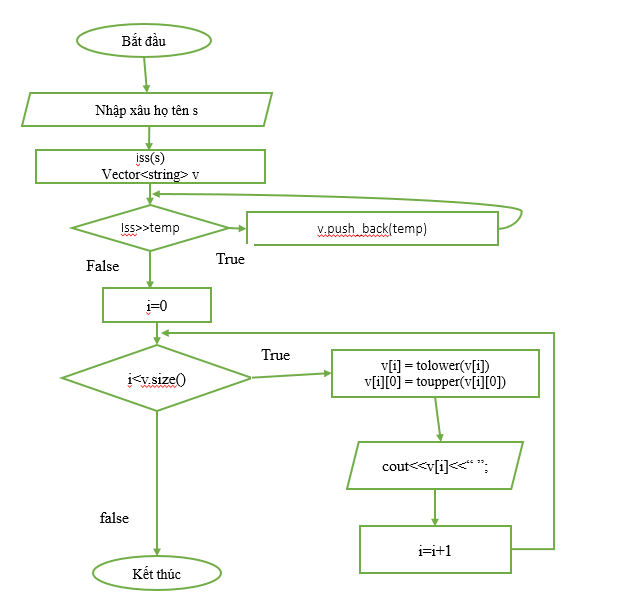
**Input:** Xâu họ tên s

**Output:** Xâu kí tự s được chuẩn hóa theo định dạng yêu cầu.

1. Phân tích bài toán.

* Xâu họ tên s được viết có thể có các khoảng trắng thừa -> sử dụng kiểu dữ liệu istringstream của c++ để lưu trữ và loại bỏ các dấu cách thừa và lấy ra được từng từ trong xâu.
* Chữ cáu đầu mỗi từ viết hoa, các chữ cái sau viết thường -> chuyển toàn bộ từ thành viết thường -> Chuyển chữ cái đầu mỗi từ thành viết hoa.
* Chuyển từng từ vừa chuẩn hóa ra màn hình và cách nhau 1 khoảng trắng.

1. Biểu diễn giải thuật

****

1. Chạy từng bước

Bước 1: Nhập xâu họ tên s;

Bước 2: Dùng kiểu dữ liệu istringstream để lưu trữ xâu họ tên và loại bỏ các dấu cách thừa.

Iss(s)

vector<string> v;

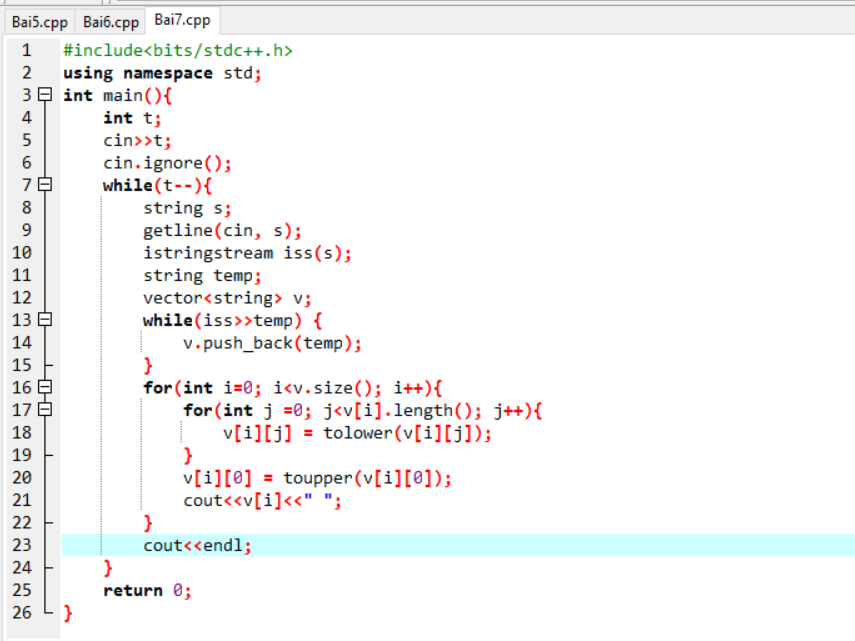
Bước 3: Lưu từng giá trị trong iss(s) vào trong vector v.

Bước 4: Chuẩn hóa từng từ trong vector v. (Chữ đầu tiên viết hoa, các chữ còn lại viết thường)

Bước 5: In kết quả ra màn hình theo đúng định dạng.

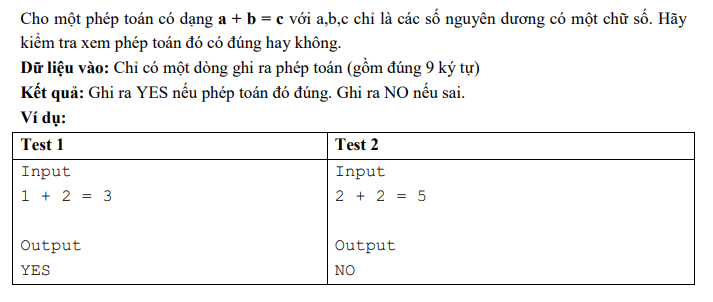
Bước 6: Kết thúc chương trình.

1. Chuẩn hóa sang code.

****

**Độ phức tạp của thuật toán: O(M+N)**

# **BÀI 5: PHÉP CỘNG**

****

1. Xác định Input/Output.

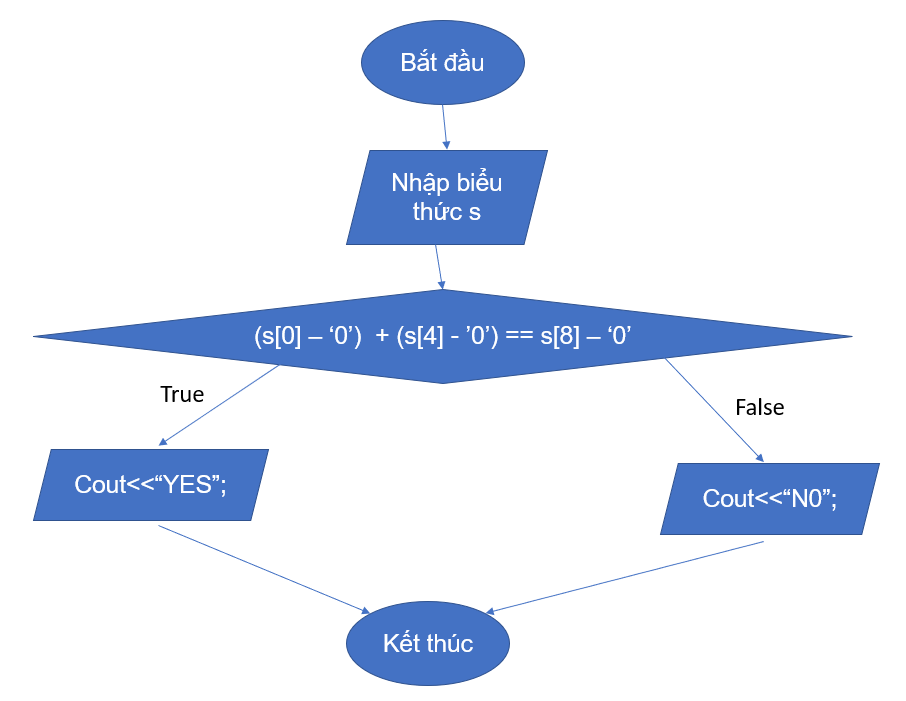
Input**:** xâu kí tự có 9 ký tự biểu diễn 1 phép toán.

Output**:** Phép toán đúng hoặc sai.

1. Phân tích bài toán. Nhập phép toán dạng xâu ký tự.

* Tách phần tử tại vị trí 0(a) và vị trí 4(b) và vị trí 8(c) và đổi thành số. Làm đầu vào và kết quả cho phép toán.
* Tính kết quả nếu a + b = c thì sẽ đưa ra kết quả là YES ngược lại là NO.

1. Biểu diễn giải thuật.



1. Chạy từng bước.

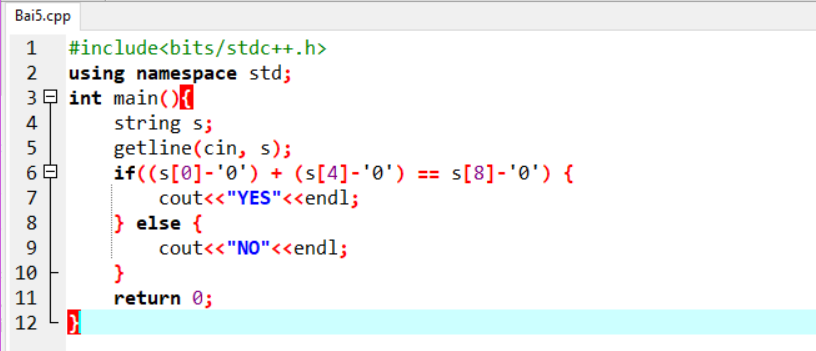
Bước 1: Nhập dữ liệu (Xâu ký tự biểu diễn phép toán)

Bước 2: So sánh tổng giá trị ở 2 vị trí 0 và 4 sau khi chuyển về dạng số với giá trị ở vị trí 8 sau khi đổi sang số.

Bước 3: In kết quả ra màn hình theo kết quả của bước 2

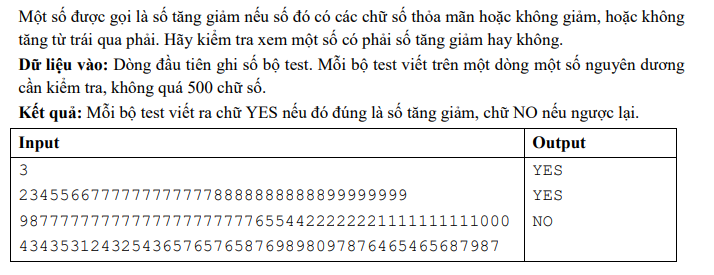
Bước 4: Kết thúc chương trình.

1. Chuyển hóa sang code:



**Đánh giá độ phức tạp của thuật toán: O(1)**

# **BÀI 6: SỐ TĂNG GIẢM**

****

1. Xác định Input/Output.

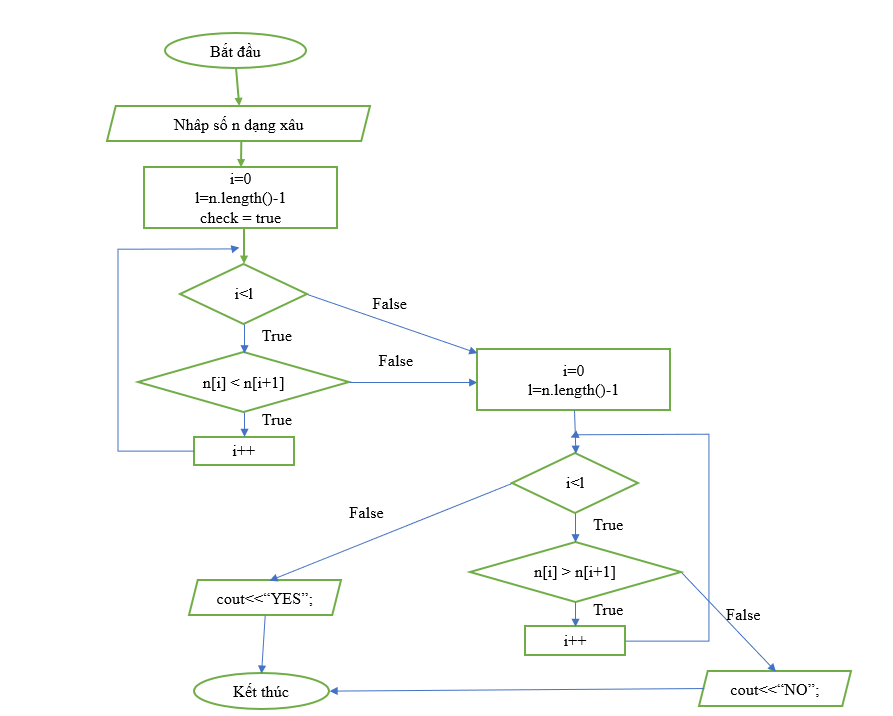
Input**:** Số nguyên dương không quá 500 chữ số.

Output**:** Số nguyên dương có phải là số tăng giảm hay không. YES/NO

1. Phân tích bài toán.

* Số nguyên dương không quá 500 chữ số -> Sử dụng xâu string để lưu trữ số nguyên dương.
* Số nguyên dương là số tăng giảm nếu nó không tăng hoặc không giảm từ trái qua phải -> lần lượt các phần từ của mảng kí tự và so sánh các giá trị của phần tử hiện tại với giá trị của phần tử tiếp theo.

1. Biểu diễn giải thuật.



1. Chạy từng bước.

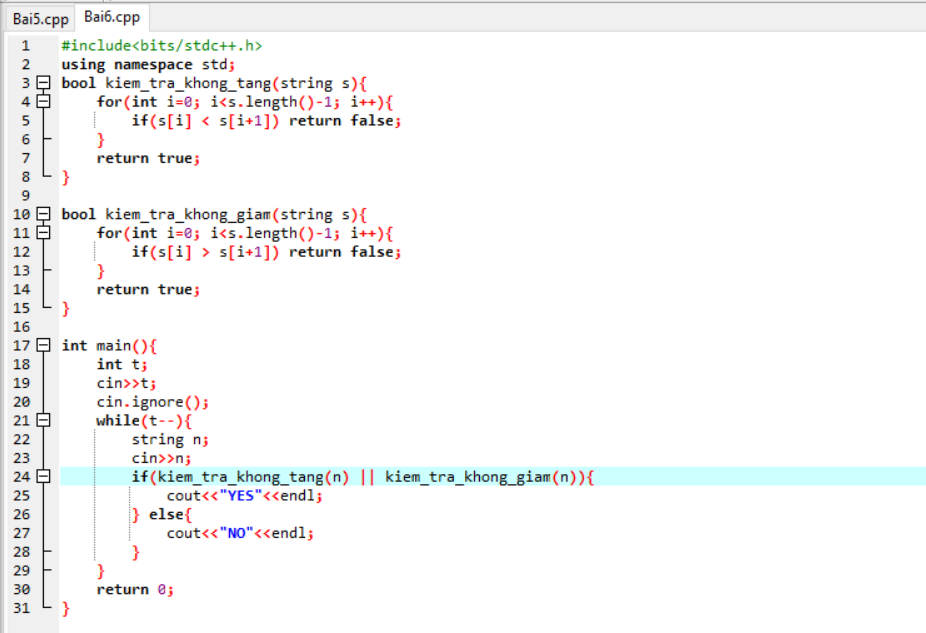
Bước 1: Nhập số n dạng xâu ký tự.

Bước 1: Viết hàm kiem\_tra\_khong\_tang(n) kiểm tra xem n có phải là số tăng hay không. (Chạy từ phần tử đầu tiên đến phần tử có vị trí n-2 và so sánh giá trị phần tử hiện tại với phần tử ngay phía sau nếu bé hơn -> return false, kết thúc hàm return true).

Bước 2: Viết hàm kiem\_tra\_khong\_giam(n) kiểm tra xem n có phải là số giảm hay không. (Chạy từ phần tử đầu tiên đến phần tử có vị trí n-2 và so sánh giá trị phần tử hiện tại với phần tử ngay phía sau nếu lớn hơn -> return false, kết thúc hàm return true).

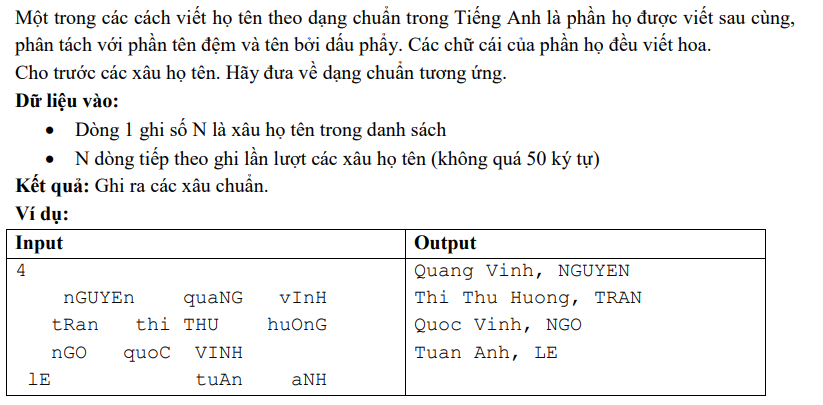
Bước 3: Kiếm tra số n qua 2 hàm kiem\_tra\_tang(n) và kiem\_tra\_giam(n) nếu 1 trong 2 hàm trả về kết quả là true thì in ra “YES” ngược lại in ra “NO”

Bước 4: Kết thúc chương trình.

1. Chuyển hóa sang code.

**Đánh giá độ phức tạp của thuật toán: O(2n) = O(n)**

# **BÀI 8: CHUẨN HÓA 2**

****

1. Xác định Input/Output.

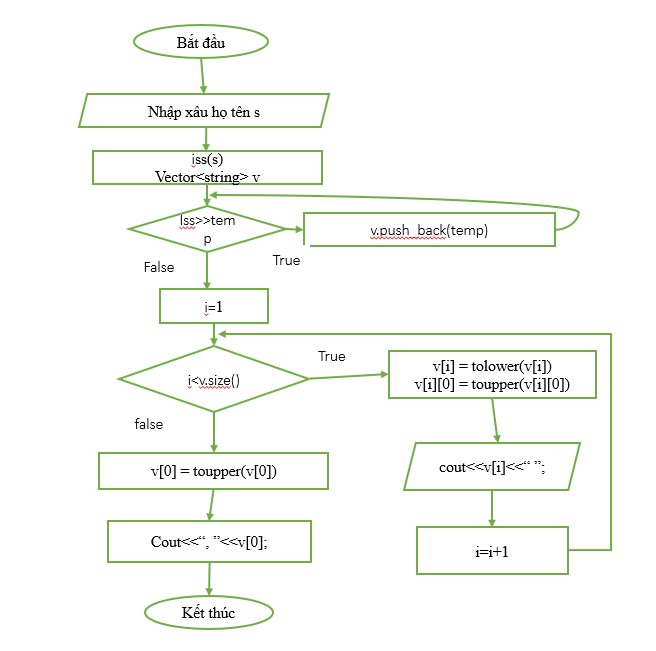
Input**:** xâu họ tên s.

Output**:** Xâu họ tên s đã được chuẩn hóa.

1. Phân tích bài toán.

* Xâu họ tên được chuẩn hóa theo dạng chuẩn tiếng anh, phần họ viết sau cùng -> Lưu xâu kí tự vào 1 mảng -> phần tử đầu tiên là phần họ sẽ in cuối cùng.
* Các chữ cái của phần họ viết hoa -> Dùng hàm toupper để viết hoa phần họ.
* Các phần còn lại chuẩn hóa theo dạng từ đầu tiên viết hoa, các từ còn lại viết thường -> dùng hàm tolower dể chuẩn hóa các từ về dạng viết thường -> dùng hàm toupper để viết hoa kí tự đầu tiên của mỗi từ.

1. Biểu diễn giải thuật.



1. Chạy từng bước.

Bước 1: Nhập xâu họ tên s.

Bước 2: Sử dụng kiểu istringstream để lưu các từ trong s.

iss(s), vector<string> v;

Bước 3: Lưu các từ trong iss(s) vào trong vector v.

Bước 4: Lặp các từ từ vị trí thứ 1 -> n-1 trong vector v để thực hiện chuẩn hóa. (Chữ cái đầu tiên viết hoa, các chữ còn lại viết thường)

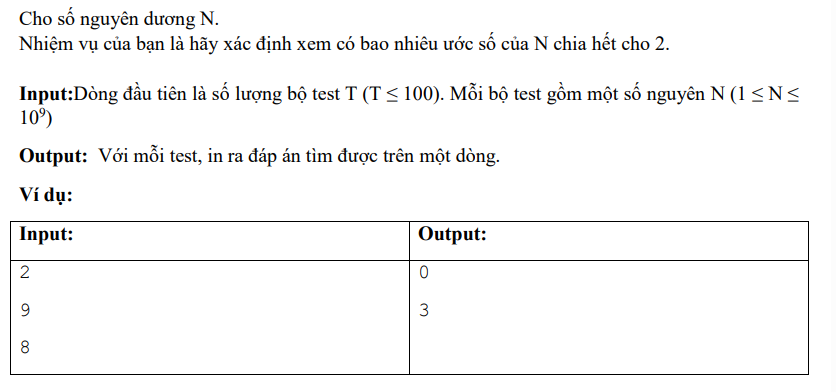
Bước 5: Chuẩn hóa phần tử đầu tiên. (các chữ đều viết hoa)

Bước 6: In kết quả ra màn hình theo đúng định dạng.

Bước 7: Kết thúc chương trình.

1. Chuyển hóa sang code.**Đánh giá độ phức tạp của thuật toán O(M+N)**

# **BÀI 10: CHIA HẾT CHO 2**

****

1. Xác định Input/Output.

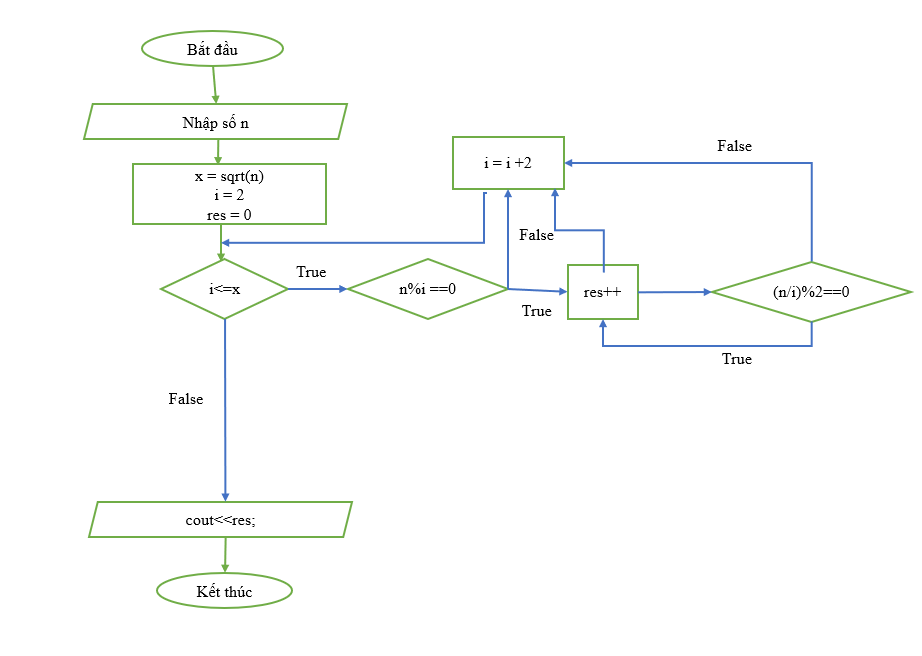
Input: Số nguyên dương N.

Output: Số ước của N chia hết cho 2.

1. Phân tích bài toán.

* Đếm có bao nhiêu ước của N chia hết cho 2.
* Nếu a là ước của n thì n/a cũng là ước của n.
* Chỉ cần chạy vòng lặp i: 2 -> -> Kiểm tra i và n/i
* Kiểm tra các ước của 2 có chia hết cho 2 không nếu có thì tăng kết quả lên +1.
* Kiểm tra n/i có chia hết cho 2 không.

1. Biểu diễn giải thuật.

****

1. Chạy từng bước

Bước 1: nhập số nguyên n

Bước 2: xác đinh các điều kiện cho vòng lặp. x=, i=2, res=0

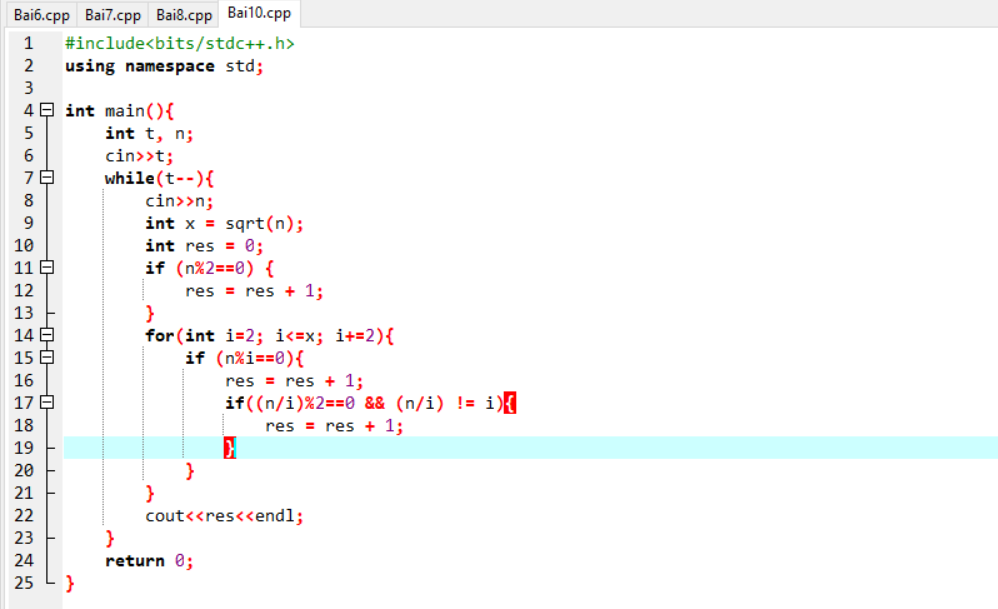
Bước 3: Vòng lặp từ i từ 2 -> x

* Nếu n%i == 0 -> res = res + 1 -> nếu (n/i)%2==0 -> res = res + 1
* i = i + 2

Bước 4: in ra kết quả res.

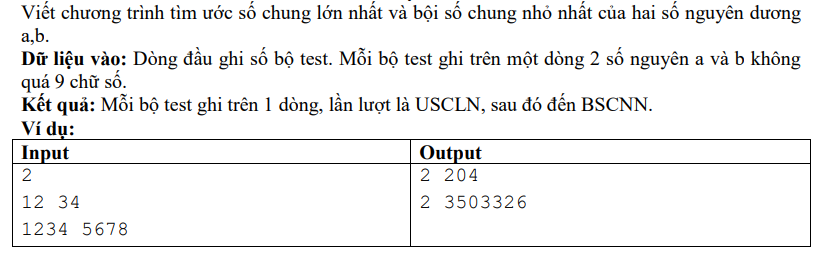
Bước 5: Kết thúc chương trình.

1. Chuyển hóa sang code.



**Đánh giá độ phức tạp của thuật toán: O(log(N))**

# **BÀI 1: ƯỚC SỐ CHUNG LỚN NHẤT VÀ BỘI CHUNG NHỎ NHẤT**

****

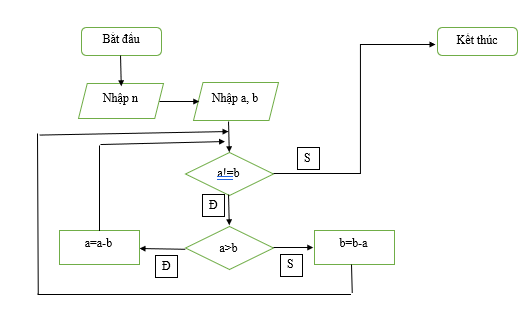
1. Input, Output:

* Input: 2 số nguyên a và b
* Output: ước chung lớn nhất và bội chung nhỏ nhất của 2 số a và b

1. Phân tích bài toán:

* Ước chung lớn nhất: Khi a != b thì lấy số lớn hơn trừ cho số bé hơn sau đó gán lại giá trị bằng hiệu vừa tính được. Khi hai số bằng nhau thì đó chính là ước chung lớn nhất
* Bội chung nhỏ nhất: Lấy tích 2 số nguyên a và b chia cho ước chung lớn nhất

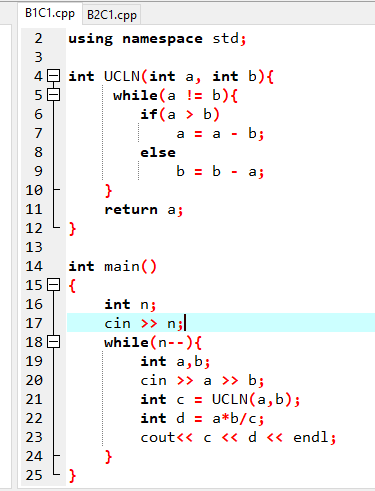
1. Biểu diễn giải thuật:



1. Chạy từng bước:

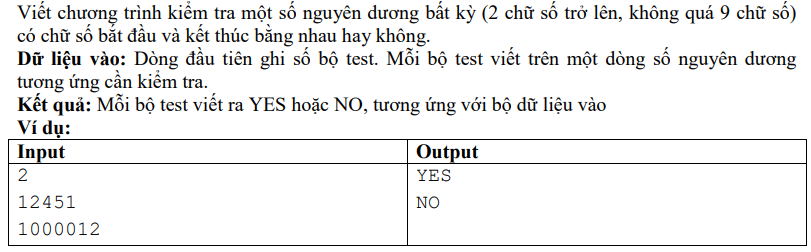
* Bước 1: Nhập số n. Mỗi lần nhập 1 cặp a, b thì n=n-1. Khi n=0 thì dừng
* Bước 2: Nhập số 2 số nguyên a và b
* Bước 3: Viết hàm tính ước chung lớn nhất
* Bước 4: Từ kết quả ước chung lớn nhất tính được bội chung nhỏ nhất
* Bước 5: Trả về kết quả

1. Code



**Độ phức tạp của thuật toán O(Max(a, b))**

# **BÀI 2: BẮT ĐẦU VÀ KẾT THỨC**

****

1. Input, Output:

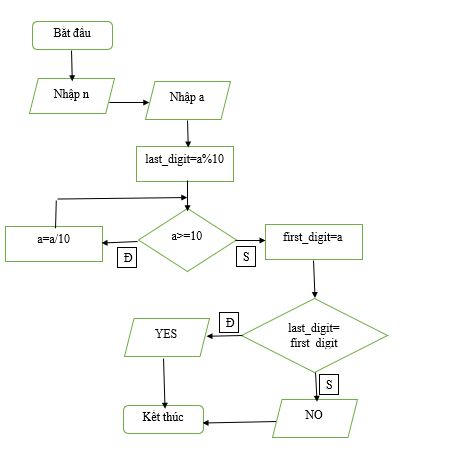
* Input:

số nguyên dương a

* Output:

chữ số đầu cuối bằng nhau in ra YES, nếu không in ra NO

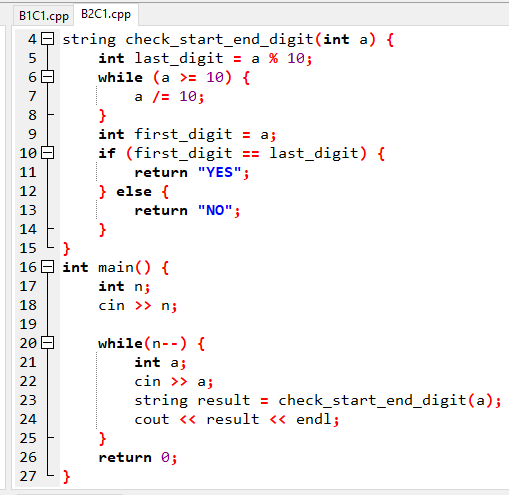
1. Phân tích bài toán: Sử dụng chia dư để lấy chữ số cuối cùng của số a, sau đó lấy phần còn lại của a chia cho 10 nếu a còn lớn hơn hoặc bằng 10, kết quả thu được sẽ là số đầu tiên. Đem so sánh 2 số và trả về kết quả
2. Biểu diễn giải thuật



1. Chạy từng bước:

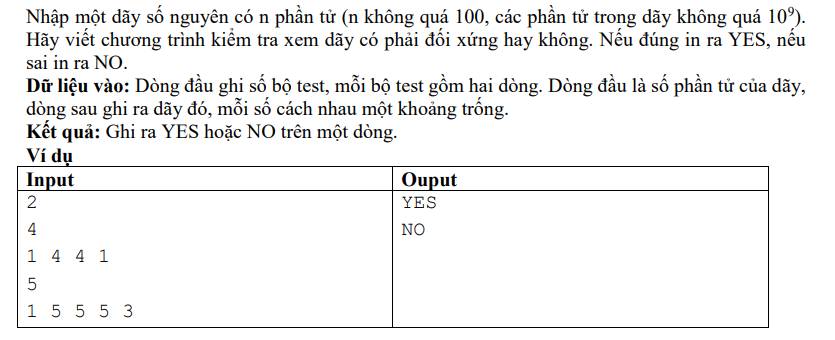
* Bước 1: Nhập số n. Mỗi lần nhập a thì n=n-1. Khi n=0 thì dừng
* Bước 2: Nhập số nguyên a
* Bước 3: Tính toán lấy ra chữ số cuối cùng và chữ số đầu tiên của a
* Bước 4: So sánh 2 số
* Bước 5 : Trả về kết quả YES hoặc NO

1. Code



**Đánh giá độ phức tạp của thuật toán: O(N), N=9**

# **BÀI 3: MẢNG ĐỐI XỨNG**

****

1. Input, Output:

* Input:

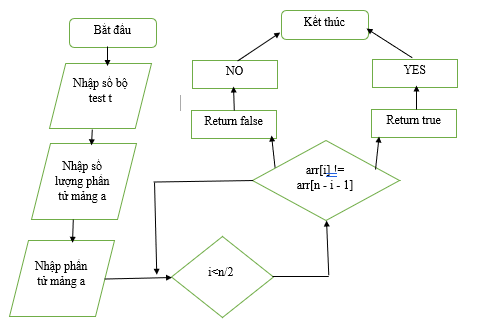
n số phần tử của mảng

các phần tử trong mảng

* Output:

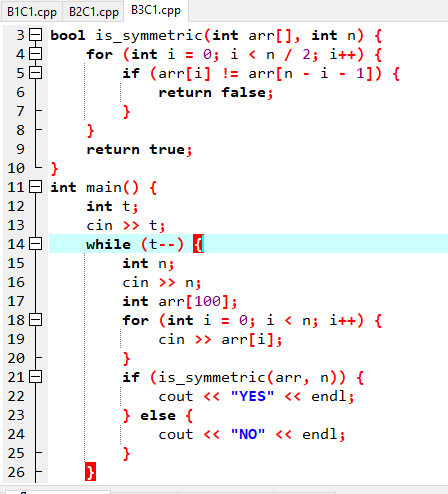
nếu mảng đối xứng in ra YES, nếu không in ra NO

1. Phân tích bài toán: Mảng được coi là đối xứng khi phần tử đầu tiên vị trí 0 bằng phần tử cuối cùng vị trí n-1, phần tử thứ 2 vị trí 1 bằng phần tử vị trí n-2,...Ý tưởng là lặp để so sánh các phần từ trong mảng từ vị trí i đến vị trí n-i-1, nếu trả về true in ra YES, nếu không in ra NO
2. Biểu diễn giải thuật:

****

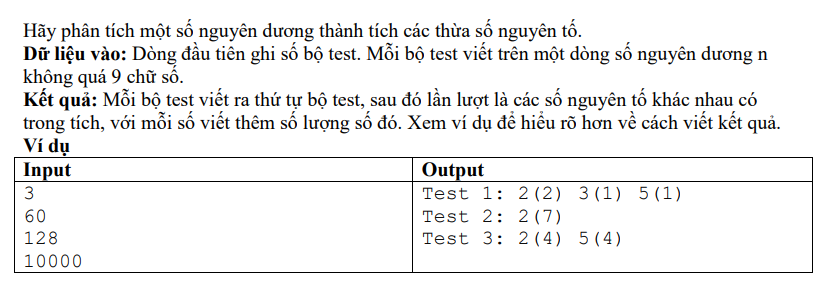
1. Chạy từng bước:

* Bước 1: Nhập số bộ test t. Mỗi lần nhập mảng a thì t=t-1. Khi n=0 thì dừng
* Bước 2 : Nhập n là số lượng phần tử mảng a
* Bước 3: Nhập các phần tử mảng a
* Bước 4: Tạo hảm boolean check điều kiện a[i] = a[n-i-1]
* Bước 5 : Nếu hàm boolean return true in ra YES, return false in ra No

1. Chuyển hóa sang code. 

**Đánh giá độ phức tạp của thuật toán: O(N/2) = O(LOG(N))**

# **BÀI 4: PHÂN TÍNH THỪA SỐ NGUYÊN TỐ.**

****

1. Input, Output:

* Input:

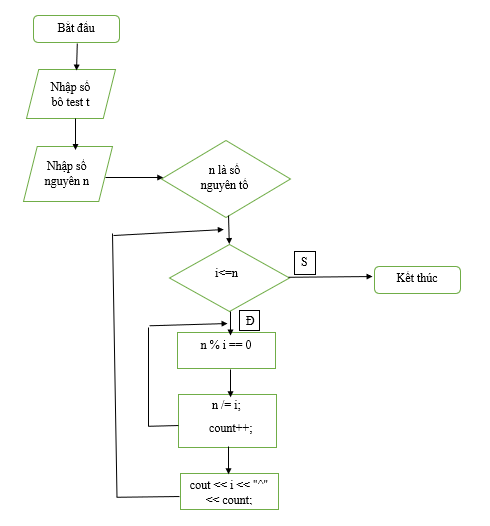
Nhập số nguyên dương n

* Output:

Các thừa số nguyên tố khác nhau

Số lượng từng thừa số nguyên tố

1. Phân tích bài toán: Trước hết ta phải kiểm tra số đã nhập có phải số nguyên tố hay không. Sau khi biết số đã nhập là số nguyên tố, ta sẽ phân tích số này thành các thừa số nguyên tố: chia số đã nhập cho i, i bắt đầu từ 2 đến n, nếu n%i==0 thì i là thừa số nguyên tố của n, đánh dấu để thừa số i không bị lặp lại nữa. Tiếp tục chia n cho i đến khi n%i!=0 thì dừng lại, ta có số lượng thừa số i.
2. Biểu diễn giải thuật:



1. Chạy từng bước:

* Bước 1: Nhập số bộ test t. Mỗi lần nhập số a thì t=t-1. Khi t=0 thì dừng
* Bước 2: Nhập số nguyên n
* Bước 3: Kiểm tra n có phải số nguyên tố không
* Bước 4: Phân tích thừa số nguyên tố của số nguyên tố n
* Bước 5: Trả về kết quả

1. Code

