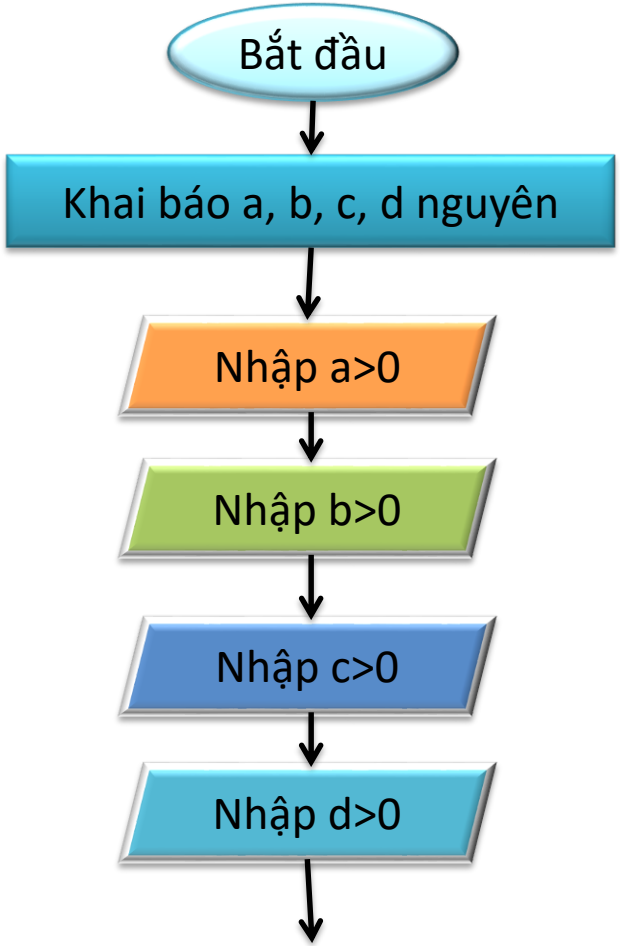
Xét bài toán

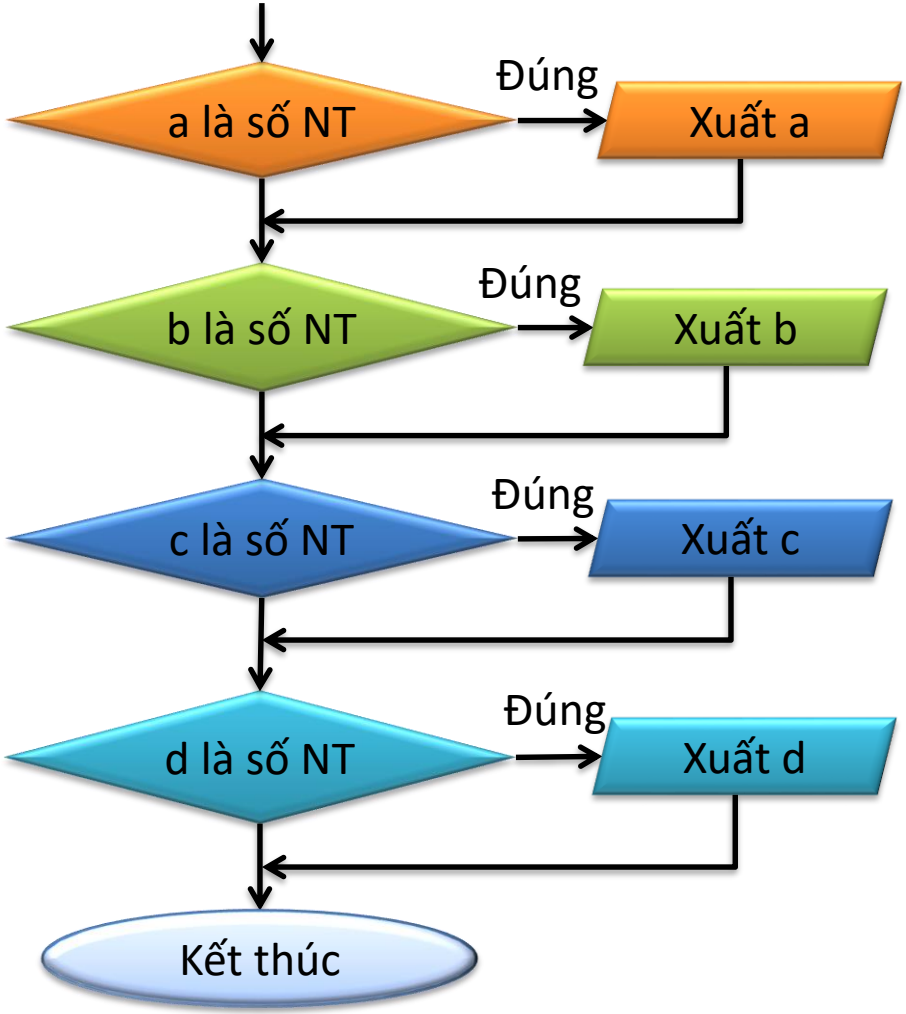
- Hãy viết chương trình thực hiện:
  - Nhập 4 số nguyên dương a, b, c, d từ bàn phím
  - Xuất ra các số nguyên tố trong những số vừa nhập

⇒ Xác định yêu cầu:

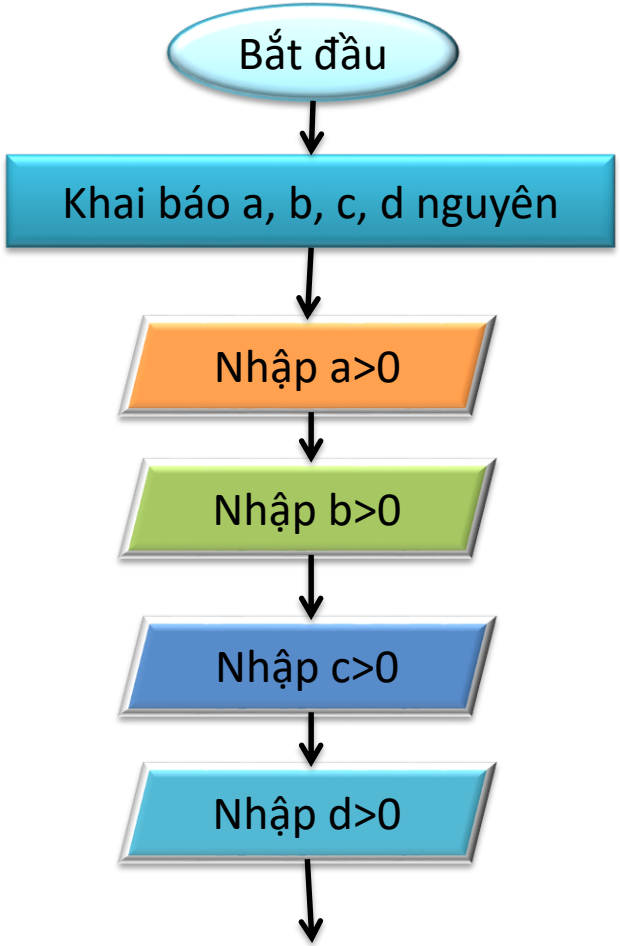
- Đầu vào: các số **nguyên**, **dương**
- Đầu ra: các số **nguyên tố**

Giải thuật

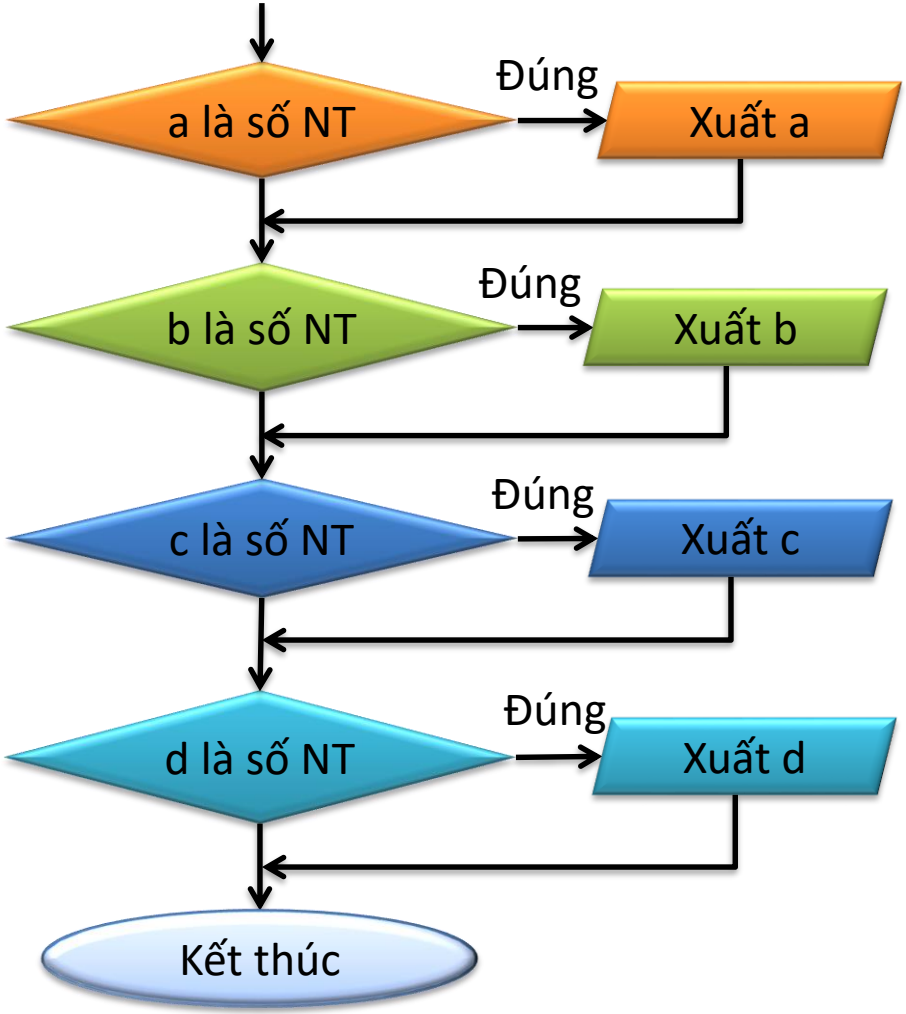



a là số NT
Đúng
Xuất a
b là số NT
Đúng
Xuất b
c là số NT
Đúng
Xuất c
d là số NT
Đúng
Xuất d
Kết thúc

Giải thuật




```
do
{
    cout<<"Nhap so nguyen duong d: ";
    cin>>d;
}while (d<0);
```




```
dem = 0;
for(int i = 2; i<= d/2; i++)
{
    if(d % i == 0)
    {
        dem++;
        break;
    }
}
if(dem==0)
    cout<< d <<" ";
```

```
dem = 0;
for(int i = 2; i<= a/2; i++)
{
    if(a % i == 0)
    {
        dem++;
        break;
    }
}
if(dem==0)
    cout<< a <<" ";
```

```
dem = 0;
for(int i = 2; i<= b/2; i++)
{
    if(b % i == 0)
    {
        dem++;
        break;
    }
}
if(dem==0)
    cout<< b <<" ";
```

```
dem = 0;
for(int i = 2; i<= c/2; i++)
{
    if(c % i == 0)
    {
        dem++;
        break;
    }
}
if(dem==0)
    cout<< c <<" ";
```

```
dem = 0;
for(int i = 2; i<= d/2; i++)
{
    if(d % i == 0)
    {
        dem++;
        break;
    }
}
if(dem==0)
    cout<< d <<" ";
```

TRƯỜNG ĐẠI HỌC THỦY LỢI
KHOA CÔNG NGHỆ THÔNG TIN
BỘ MÔN TIN HỌC VÀ KTTT

LẬP TRÌNH NÂNG CAO
HÀM

Giảng viên: Nguyễn Thị Phương Dung
Email: dungntp@tlu.edu.vn

Nội dung bài học

- Khái niệm hàm
- Cách khai báo hàm
- Cách sử dụng hàm
- Cách truyền tham số vào hàm

Khái niệm Hàm

- Hàm là một phần của chương trình, dùng để giải quyết một công việc nhỏ trong một bài toán lớn.
- Một bài toán lớn có thể được chia thành nhiều bài toán nhỏ (nhiều hàm) để dễ quản lý
- Hàm có thể có hoặc không có giá trị trả về

Khái niệm Hàm

- Mọi khái niệm như thủ tục, chương trình con trong các ngôn ngữ lập trình khác đều được hiểu là hàm trong C++
- Mọi chương trình C++ đều có 1 hàm chính là hàm main()

Cấu trúc chương trình khi viết hàm

- Cách 1:

  Khai báo kết hợp định nghĩa hàm

```
1  //phan khai bao thu vien
2  #include<iostream>
3  using namespace std;
4
5  //phan khai bao va dinh nghia ham
6
7  //ham chinh
8  int main()
9  {
10     //loi goi ham
11     return 0;
12 }
```

Cấu trúc chương trình khi viết hàm

- Cách 2:

  Khai báo nguyên mẫu hàm trước

  Định nghĩa hàm sau

```
1  //phan khai bao thu vien
2  #include<iostream>
3  using namespace std;
4
5  //phan khai bao nguyen mau ham
6
7  //ham chinh
8  int main()
9  {
10     //loi goi ham
11     return 0;
12 }
13
14 //phan dinh nghia ham
15
16
```

Cú pháp khai báo hàm

- Với cách 1: Khai báo kết hợp định nghĩa hàm

```
kieutrave tenham(kieuthamso1 ts1, kieuthamso2 ts2,..)
{
    //các lệnh xử lý
    return giatri; //giá trị trả về phải có cùng kiểu với kieutrave của hàm
}
```

Cú pháp khai báo hàm

- Với cách 2: Khai báo trước, định nghĩa sau
- Khai báo:

```
kieutrave tenham(kieuthamso1 ts1, kieuthamso2 ts2,...);
kieutrave tenham(kieuthamso1, kieuthamso2,...);
```

- Định nghĩa:

```
kieutrave tenham(kieuthamso1 ts1, kieuthamso2 ts2,...)
{
    //các lệnh xử lý
    return giatritrave; //giá trị trả về phải cùng kiểu
                        //với kieutrave của hàm
}
```

Quy ước khi khai báo hàm

- **kieutrave** và các **kieuthamso** thường là các kiểu dữ liệu cơ bản
- Hàm không có giá trị trả về thì **kieutrave** là **void**
- **tenham** và tên các tham số **ts1**, **ts2**: *đặt theo quy tắc định danh*
- *Hàm không có tham số đầu vào thì viết:* ***tenham()***

Quy ước khi khai báo hàm

- Lệnh **return:**
  - *Trả về giá trị cho hàm (có thể trả về giá trị của cả biểu thức)*
  - *Có thể xuất hiện lênh* return *nhiều lần trong hàm*
  - *Có tác dụng kết thúc hàm*
  - *Hàm* ***void*** *không cần lệnh* ***return***

Ví dụ cách khai báo và định nghĩa hàm

```
int nhapsoduong()
{
    int t;
    do
    {
        cout<<"Nhap so nguyen duong: ";
        cin>>t;
    }while (t<0);
    return t;
}
```

Kiểu trả về

Tên hàm

Hàm không có tham số nên chỉ cần viết **()**

Lệnh return trả về giá trị có cùng kiểu trả về của hàm

Ví dụ cách khai báo và định nghĩa hàm

```
void nhapsoduong(int &t)
{
    do
    {
        cout<<"Nhap so nguyen duong: ";
        cin>>t;
    }while (t<0);
}
```

Kiểu trả về là **void**

Hàm có tham số nên các tham số được viết trong cặp ngoặc **()**

Nên cuối hàm không có lệnh **return**

Sử dụng hàm

- Với hàm có giá trị trả về có thể sử dụng theo các cách sau:
  - Gán giá trị cho biến: tenbien = tenham(ts1, ts2, …);
    - VD:
```
a = nhapsoduong();
```
  - Xuất kết quả của hàm ra màn hình thông qua std::cout:
    cout << tenham(ts1, ts2, …);
    - VD:
```
cout<<"So nho nhat la: "<<min(a,b);
```
  - Sử dụng như một biểu thức logic: if(tenham(ts1, ts2, …))
    - VD:
```
if(nguyento(a))
```

Sử dụng hàm

- Với hàm không có giá trị trả về (void fx()) thì:
  - Chỉ có thể gọi trực tiếp để thực hiện các nhiệm vụ
    - VD: fx(); ☑
  - Không được gọi trong lệnh xuất dữ liệu ra màn hình:
    - VD: ~~cout<<fx();~~
  - Không được dùng để gán giá trị cho biến
    - VD: ~~int i = fx();~~

Ví dụ sử dụng hàm

Hàm có kiểu trả về là **void**

Hàm có tham số

Chỉ có thể gọi hàm thực hiện như một lệnh

Khi gọi hàm phải truyền tham số cùng kiểu với tham số trong khai báo hàm

```
int nhapsoduong();
void nhapsoduong(int &t);
int main()
{
    int a, b;
    a = nhapsoduong();
    nhapsoduong(b);
    return 0;
}
```

Tiện ích khi sử dụng hàm

- Thay vì phải viết nhiều lần một đoạn mã lệnh thì ta chỉ cần viết một lần và khi nào cần dùng thì gọi nó ra.

Các cách truyền tham số vào hàm

Truyền tham trị

- Truyền bản sao giá trị của biến vào hàm
- => hàm làm việc với bản sao của biến

Truyền tham chiếu

- Truyền địa chỉ của biến vào hàm
- => hàm làm việc trực tiếp với biến (bản gốc)

Các cách truyền tham số vào hàm

Truyền tham trị

- Sau khi ra khỏi hàm, giá trị của biến (bản gốc) không bị thay đổi

Truyền tham chiếu

- Sau khi ra khỏi hàm, giá trị của biến (bản gốc) bị thay đổi

Sử dụng cách truyền tham số nào?

Truyền tham trị

- Khi không cần lấy ra sự thay đổi của tham số truyền vào

Truyền tham chiếu

- Khi cần lấy ra giá trị thay đổi của tham số

Chú ý khi khai báo hàm

- Tham số nào cần truyền theo kiểu tham chiếu thì phải có dấu **&** trước tên tham số đó

Ví dụ các cách truyền tham số

- Truyền tham trị

Hàm khai báo cần truyền vào 1 số nguyên

Khi gọi hàm sẽ truyền vào biến cần làm việc, nhưng hàm sẽ làm việc trên bản sao của biến

```
bool ktraNT(int x)
{
    for(int i = 2; i<= x/2; i++)
    {
        if(x % i == 0)
            return false;
    }
    return true;
}
int main()
{
    int a = 5, b = 6;
    if(ktraNT(a))
        cout<<a<<" la so nguyen to"<<endl;
    return 0;
}
```

Ví dụ các cách truyền tham số

- Truyền tham chiếu

Khi khai báo cần có dấu & trước tên biến để chỉ ra rằng cần truyền vào địa chỉ của biến

Khi gọi hàm vẫn truyền vào biến cần làm việc, nhưng hàm sẽ làm việc trên bản gốc của biến

```
void nhapsoduong(int &t)
{
    do
    {
        cout<<"Nhap so nguyen duong: ";
        cin>>t;
    }while (t<0);
}
int main()
{
    int a;
    nhapsoduong(a);
    cout<<"So vua nhap la: "<<a;
    return 0;
}
```

Câu hỏi

- Hãy cho biết kết quả của đoạn chương trình sau?
- Với dữ liệu ban đầu:

  a = 5, b = 9
- => kết quả:

  a = 5, b = 9

```
int fx(int x, int y)
{
    int t = 0;
    t = 5*x + 3*y + 2;
    x++;
    y--;
    return t;
}
int main()
{
    int a, b;
    cout<<"Nhap a, b: "; cin>>a>>b;
    fx(a, b);
    cout<<"a = "<<a<<endl;
    cout<<"b = "<<b<<endl;
    return 0;
}
```

Câu hỏi

- Hãy cho biết kết quả của đoạn chương trình sau?
- Với dữ liệu ban đầu:

  a = 5, b = 9
- => kết quả:

  a = 10, b = 8

```
int fy(int &x, int &y)
{
    int t = 0;
    t = 5*x + 3*y + 2;
    x += 5;
    y--;
    return t;
}
int main()
{
    int a, b;
    cout<<"Nhap a, b: "; cin>>a>>b;
    fy(a, b);
    cout<<"a = "<<a<<endl;
    cout<<"b = "<<b<<endl;
    return 0;
}
```

Bài tập

- Áp dụng các cách viết hàm để giải bài toán:
  - Nhập 4 số nguyên dương a, b, c, d từ bàn phím
  - Xuất ra các số nguyên tố trong những số vừa nhập

Đệ quy

Đệ quy

- Là một phương pháp lập trình cho phép một hàm có thể gọi lại chính nó trực tiếp hoặc gián tiếp.
- Một chương trình đệ quy hoặc một định nghĩa đệ quy thì không thể gọi đến chính nó mãi mãi mà phải có một điểm dừng đến một trường hợp đặc biệt nào đó, đó là trường hợp suy biến (degenerate case).

Ví dụ

- Viết hàm tính n! (n giai thừa)

```
int giaithua(int n){
    if(n<0)//loi => thoat chuong trinh
        exit(1);
    if(n==0)//diem dung
        return 1;
    return n*giaithua(n-1);//goi de quy
}
```

Ví dụ

- Viết hàm tính x^n (x mũ n)

```
int hammu(int x, int n){
    if(n<0)
        return 1/x*hammu(x, n+1);
    if(n == 0)
        return 1;
    return x*hammu(x,n-1);
}
```

Đệ quy

- Chương trình đệ quy gồm hai phần chính:
  1. Phần cơ sở: Điều kiện thoát khỏi đệ quy (điểm dừng)
  2. Phần đệ quy: Trong phần thân chương trình có lời gọi đến chính bản thân chương trình với giá trị mới của tham số nhỏ hơn giá trị ban đầu

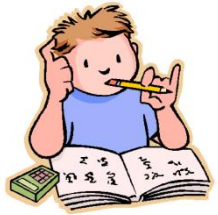
Phương pháp thiết kế một giải thuật đệ quy

- Phân rã bài toán thành các bài toán nhỏ hơn.
- Trong đó có ít nhất một nhiệm vụ con là một trường hợp nhỏ hơn của nhiệm vụ cha (giống hệt bài toán ban đầu)
- Phân tích cho đến khi bài toán đủ nhỏ để cho ra kết quả đúng => đó là điểm dừng

Chú ý

- Nếu không thiết kế điểm dừng, hàm đệ quy đó sẽ trở thành đệ quy vô hạn

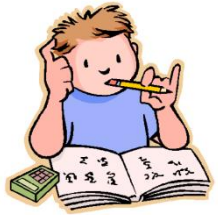
Bài tập



- Viết hàm đệ quy tìm ước số chung lớn nhất của 2 số nguyên a và b

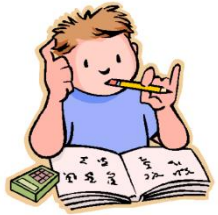
```
int ucln(int a, int b)
{
    if(a%b == 0)
        return b;
    if(b%a == 0 )
        return a;
    if(a>b) return ucln(a-abs(b), b);
    else    return ucln(b-abs(a), a);
}
```

Bài tập



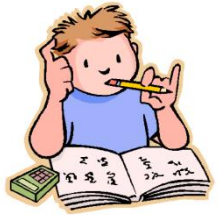
- Viết hàm đệ quy tính số Fibonaci thứ n theo công thức:
  - f(n) = f(n-1) + f(n-2) với n>=3
  - f(n) = 1 với n=1 hoặc n = 2
- Viết chương trình in ra dãy n số Fibonaci

Bài tập



- Viết hàm đệ quy cho phép in ra các chữ số của một số nguyên dương theo chiều xuôi, mỗi chữ số cách nhau một dấu cách (VD: cho số 57294 thì in ra 5 7 2 9 4)
- Viết hàm đệ quy cho phép in ra các chữ số của một số nguyên dương theo chiều ngược, mỗi chữ số cách nhau một dấu cách (VD: cho số 57294 thì in ra 4 9 2 7 5)

Bài tập



- Viết hàm đệ quy cho phép in ra dạng nhị phân của một số nguyên

Nạp chồng hàm

Nạp chồng hàm

- Nạp chồng hàm là cách tạo ra những hàm
  - Giống nhau về tên hoặc
  - Giống nhau về cả tên và kiểu trả về
  - Nhưng phải khác nhau về kiểu tham số hoặc
  - Khác nhau về số các tham số

Ví dụ

Nạp chồng hàm có cùng tên là **max**, cùng kiểu trả về là **int**, khác với hàm có sẵn trong thư viện là có **3 tham số**

Nạp chồng hàm có cùng tên là **max**, cùng số lượng các tham số là 2 so với hàm có sẵn trong thư viện nhưng khác kiểu trả về là **string**

Trong hàm main() sử dụng 2 hàm được nạp chồng cùng hàm có sẵn trong thư viện

```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  int max(int a, int b, int c)
6  {
7      int m;
8      m = (a>b) ? a : b;
9      m = (m>c) ? m : c;
10     return m;
11 }
12 string max(string a, string b)
13 {
14     return (a.length() > b.length()) ? a : b;
15 }
16 int main()
17 {
18     cout << "max(1, 20) = " << max(1, 20) << endl;
       cout << "max(9, 7, 10) = " << max(9, 7, 10) <<endl;
       cout << "max(hello, xin chao ban) = " << max("hello", "xin chao ban");
21     return 0;
22 }
```