**HÀM**

* Tên hàm: động từ, đúng ý nghĩa
* Tham số: input (trị, biến)/output (biến)
* Kiểu trả về: output
* Lưu ý: Tính tái sử dụng của hàm

**Ví dụ 1:** *Viết hàm nhập vào số nguyên dương (nhập đến khi nào đúng thì dừng)*

**Cách phân tích 1:**

Tên hàm: *nhapSoDuong*

Tham số: tham biến *soDuong* (số cần nhập) kiểu số nguyên

Kiểu trả về: *không có*

void nhapSoDuong(int soDuong)

{

do{

cout<< “Nhap so nguyen duong:”;

cin>>soDuong;

}while(soDuong<=0);

}

Chạy thử:

void main()

{

int n;

nhapSoDuong(n);

cout<< “So duong vua nhap:”<<n;

}

**Cách phân tích 2:**

Tên hàm: *nhapSoDuong*

Tham số: không có

Kiểu trả về: *số nguyên vừa nhập*

int nhapSoDuong()

{

int soDuong;

do{

cout<< “Nhap so nguyen duong:”;

cin>>soDuong;

}while(soDuong<=0);

return soDuong;

}

Chạy thử:

void main()

{

int n=nhapSoDuong();

cout<< “So duong vua nhap:”<<n;

}

**Ví dụ 2:** Viết hàm kiểm tra số hoàn thiện.

Số hoàn thiện là số tự nhiên mà **có** **tổng các ước số dương** của nó **bé hơn** nó **bằng** chính nó. Ví dụ: số 6 có các ước số dương bé hơn nó (<6) là 0,1,2,3 và tổng là 0+1+2+3=6 nên 6 là số hoàn thiện.

**Cách phân tích 1:**

Tên hàm: *isPerfectNumber*

Tham số: tham trị *number* (số cần kiểm tra) kiểu số nguyên, *vì khi sử dụng hàm này chỉ muốn kiểm tra số là hoàn thiện chứ không muốn thay đổi giá trị của số.*

Kiểu trả về: *bool (true là số hoàn thiện, false không là số hoàn thiện)*

bool *isPerfectNumber* (int number)

{

int sum=0;

for(int i=1; i<number; i++)

{

if(number%i==0)

{

sum+=i;

}

}

return (sum==number);

}

Chạy thử:

void main()

{

int n=nhapSoDuong();

if(isPerfectNumber(n)==true)

{

cout<< “La so hoan thien”;

}

else

{

cout<< “Khong la so hoan thien”;

}

}

**HÀM ĐỆ QUY**

Đệ quy: ***Bước sau được tính từ 1 hay nhiều bước trước đó***

Vòng lặp vs Hàm đệ quy:???  
**Ví dụ 1:** Tính phần tử un của dãy số Fibonasi, biết:



**Cách 1:** Sử dụng vòng lặp 🡪 biến tạm lưu trữ kết quả bước trước

int U(int n)

{

int **U\_n\_2**=1;

int **U\_n\_1**=1;

int U\_n=1;

for(int i=2;i<=n;i++)

{

U\_n = U\_n\_1+ U\_n\_2;

**U\_n\_2 = U\_n\_1;**

**U\_n\_1 = U\_n;**

}

return U\_n;

}

**Cách 2:** Sử dụng đệ quy

int U(int n)

{

if(n==0 || n==1) return 1;

return **U(n-1)+U(n-2);**

}

Chạy từng bước:U(3)