



28TECH

Become A Better Developer



HÀM (FUNCTION)



KHÁI QUÁT VỀ HÀM



Phần lớn các chương trình máy tính được xây dựng để **giải quyết các bài toán lớn** trong thực tế.



Cách tốt nhất để xây dựng và bảo trì một chương trình đó là xây dựng nó từ những **thành phần nhỏ** được xây dựng đơn lẻ.



Chức năng Hàm (Function) được sử dụng để **chia nhỏ chương trình** thành các thủ tục nhỏ giải quyết từng chức năng nhỏ



KHÁI QUÁT VỀ HÀM

Lợi ích của việc chia chương trình thành các hàm nhỏ



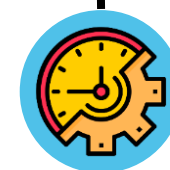
Code trở nên mạch lạc, dễ đọc



Dễ debug khi gặp lỗi



Dễ bảo trì khi cần thay đổi một chức năng nhỏ



Có khả năng tái sử dụng lại code





CÚ PHÁP (SYNTAX)

```
returntype functionName ( parameter1, parameter2,...){  
    //Function body  
}
```

TRONG ĐÓ:

returntype: Kiểu trả về của hàm (int, double, bool, void,...)

functionName: Tên hàm

parameter: Tham số của hàm

function body: Các câu lệnh bên trong của hàm





VÍ DỤ

Kiểu trả về **void**, tức hàm này **không** trả về giá trị nào

xinchao: tên hàm

Hàm này **không** có tham số

```
void xinchao () {  
    cout << "Hello 28tech!";  
}
```

GỌI HÀM (FUNCTION CALL)

```
#include <iostream>
using namespace std;
void xinchao(){
    cout << "Hello 28tech !";
}
int main(){
    cout << "Truoc khi goi ham" << endl;
    xinchao(); // Function call
    cout << "Sau khi goi ham" << endl;
}
```

Function
call



KHAI BÁO NGUYÊN MẪU HÀM (FUNCTION PROTOTYPE)



Các bạn có thể khai báo nguyên mẫu hàm ở đầu chương trình để làm cho chương trình rõ ràng hơn cũng như thông báo những hàm.



```
//Function prototype
int max_val(int, int);
//Function declaration
int max_val(int a, int b){
    if (a > b)
        return a;
    else
        return b;
}
```

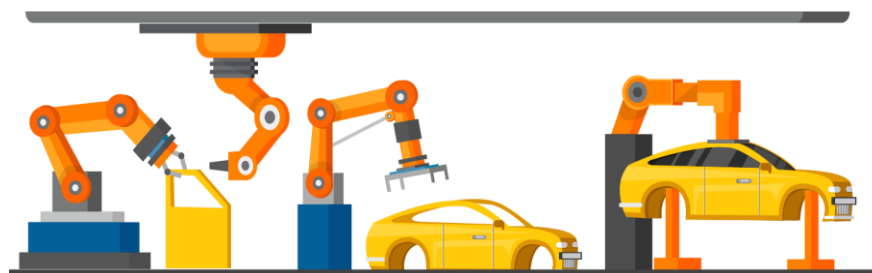


THAM SỐ VÀ GIÁ TRỊ TRẢ VỀ (PARAMETER AND RETURN TYPE)

Tham số



Hàm



Giá trị trả về



👉 Tham số là những giá trị được truyền cho hàm khi hàm được gọi.



VÍ DỤ

```
void printNumber(int a){  
    cout << a << endl;  
}  
int main(){  
    int n = 10;  
    printNumber(n);  
}
```



OUTPUT: 10

Giải thích cách hoạt động: Khi bạn gọi hàm `printNumber(n)` thì `n` được gọi là **đối số** hay **tham số thực sự**, còn `a` trong hàm là **tham số hình thức**. Giá trị của `n` sẽ được gán cho `a`, vì thế `a` cũng sẽ có giá trị là 10, và khi câu lệnh bên trong hàm `printNumber` thực hiện màn hình sẽ hiển thị giá trị 10.



VÍ DỤ

```
//Hàm nhận vào 3 số và trả về tổng  
của 3 số
```

```
int tong(int a, int b, int c){  
    return a + b + c;  
}
```

```
int main(){  
    int x = 100, y = 200, z = 300;  
    cout << tong(x, y, z) << endl;  
    tong(x, y, z);  
}
```



OUTPUT: ???





TRUYỀN THAM CHIẾU VÀ TRUYỀN THAM TRỊ

VÍ DỤ 1: Truyền tham trị

```
void add(int a){  
    a += 100;  
}  
  
int main(){  
    int n = 10;  
    add(n);  
    cout << n << endl; //10  
}
```

VÍ DỤ 2: Truyền tham chiếu

```
void add(int &a){  
    a += 100;  
}  
  
int main(){  
    int n = 10;  
    add(n);  
    cout << n << endl; //110  
}
```



TRUYỀN THAM CHIẾU VÀ TRUYỀN THAM TRỊ

VÍ DỤ 1: Truyền tham trị

Khi bạn xây dựng hàm với cách **truyền tham trị (pass by value)** thì giá trị của tham số thực sự (n) được gán cho tham số hình thức (a), và những thay đổi bên trong hàm đối với tham số hình thức a sẽ **không ảnh hưởng tới tham số thực sự**.

```
void add(int a){  
    a += 100;  
}  
  
int main(){  
    int n = 10;  
    add(n);  
    cout << n << endl; //10  
}
```





TRUYỀN THAM CHIẾU VÀ TRUYỀN THAM TRỊ

VÍ DỤ 2: Truyền tham chiếu

Ngược lại nếu bạn truyền tham chiếu (pass by reference) thì những gì bạn thay đổi trong tham số hình thức sẽ làm tham số thực sự bị thay đổi theo.

Chú ý: Nếu bạn muốn thay đổi của một tham số sau khi hàm được thực hiện xong, hãy truyền tham chiếu, ngược lại bạn hãy truyền tham trị

```
void add(int &a){  
    a += 100;  
}  
  
int main(){  
    int n = 10;  
    add(n);  
    cout << n << endl; //110  
}
```



NHỮNG CHÚ Ý KHI XÂY DỰNG HÀM





NẠP CHỒNG HÀM (FUNCTION OVERLOADING)

Nạp chồng hàm là một tính năng của C++ khi mà các hàm có cùng tên nhưng khác nhau về tham số, kiểu trả về. Khi hàm được gọi thì tùy vào kiểu dữ liệu của tham số, hàm phù hợp sẽ được gọi.

```
int findMax(int a, int b){  
    if(a > b) return a;  
    else return b;  
}  
float findMax(float a, float b){  
    if(a > b) return a;  
    else return b;  
}  
int main(){  
    int x = 100, y = 200;  
    cout << findMax(x, y) << endl;  
    float m = 100.5, n = 200.5;  
    cout << findMax(m, n) << endl;  
}
```



CÁC HÀM TOÁN HỌC PHỔ BIẾN TRONG C++



Các bạn sử dụng thư viện : `#include <bits/stdc++.h>` để thêm mọi thư viện của C++ vào chương trình, tránh phải khai báo từng thư viện riêng lẻ.

Hàm	Chức năng	Ví dụ
max	Tìm giá trị lớn nhất	<code>cout << max(10, 20); // 20</code> <code>cout << max({10, 20, 30, 40}); // 40</code>
min	Tìm giá trị nhỏ nhất	<code>cout << min(10, 20); // 10</code> <code>cout << min({10, 20, 30, 40}); // 10</code>
sqrt(n)	Hàm tính căn bậc 2 của n, trả về số double	<code>int n = 100;</code> <code>int can = sqrt(n); // can = 10.000 = 10</code>
pow (x,y)	Hàm tính x^y , trả về số double	<code>int x = 2, y = 10;</code> <code>int res = pow(x, y); // res = 1024.000 = 1024</code>
ceil(n)	Hàm trả về số nguyên lớn hơn gần nhất với n	<code>float a = 2.3;</code> <code>int res = ceil(a); // res = 3.000 = 3</code>

floor(n)	Hàm trả về số nguyên nhỏ hơn gần nhất với n	<code>float a = 2.3;</code> <code>int res = ceil(a); // res = 2.000 = 2</code>
round(n)	Làm tròn n tùy thuộc vào phần thập phân	<code>float a = 2.3;</code> <code>int res1 = round(a); // res1 = 2.000 = 2</code> <code>float b = 2.5;</code> <code>int res2 = round(b); // res1 = 3.000 = 3</code>
abs(n)	Trả về trị tuyệt đối của n	<code>cout << abs(-100); // 100</code>





28TECH

Become A Better Developer

KẾT THÚC PHẦN HÀM



FINISH

