# Bài: Nạp chồng hàm trong C++ (Function overloading)

*Xem bài học trên website để ủng hộ Kteam: Nạp chồng hàm trong C++ (Function overloading)*

*Mọi vấn đề về lỗi website làm ảnh hưởng đến bạn hoặc thắc mắc, mong muốn khóa học mới, nhằm hỗ trợ cải thiện Website. Các bạn vui lòng phản hồi đến Fanpage How Kteam nhé!*

## Dẫn nhập

Ở bài học trước, bạn đã nắm được những kiến thức về HÀM NỘI TUYẾN TRONG C++ (Inline functions).

Trong bài học này, chúng ta sẽ cùng tìm hiểu về **Nạp chồng hàm trong C++ (Function overloading)**.

---

## Nội dung

Để đọc hiểu bài này tốt nhất các bạn nên có kiến thức cơ bản về:

- CƠ BẢN VỀ HÀM VÀ GIÁ TRỊ TRẢ VỀ (Basics of Functions and Return values)

Trong bài ta sẽ cùng tìm hiểu các vấn đề:

- Đặt vấn đề
- Nạp chồng hàm trong C++ (Function overloading)
- Một số hàm không thể nạp chồng trong C++

---

## Đặt vấn đề

Chúng ta có hàm tính tổng 2 số nguyên:

:

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

Vì hàm trên có tham số **kiểu số nguyên** (**int**), nếu chúng ta có nhu cầu tính tổng 2 số chấm động, hàm này sẽ không thực hiện được.

Để giải quyết vấn đề này, chúng ta có thể tạo ra 2 hàm có tên khác nhau:

:

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

double addDouble(double a, double b)
{
	return a + b;
}
```

Như vậy, ta sẽ có nhiều hàm với các tên gọi khác nhau. Tuy nhiên, việc sử dụng tên như vậy sẽ gây bất lợi cho người lập trình khi gọi hàm. **Nạp chồng hàm (function overloading)** ra đời để giải quyết vấn đề trên.

---

## Nạp chồng hàm trong C++ (Function overloading)

**Nạp chồng hàm (function overloading)** là tính năng của ngôn ngữ C++ (không có trong C). Kỹ thuật này cho phép sử dụng **cùng một tên** gọi **cho nhiều hàm** (có cùng mục đích). Nhưng khác nhau về **kiểu dữ liệu tham số** hoặc **số lượng tham số**.

Giải quyết vấn đề trên:

:

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

double add(double a, double b)
{
	return a + b;
}
```

Chúng ta cũng có thể định nghĩa các hàm add() với số lượng tham số khác nhau:

:

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

> **Chú ý:** Việc quyết định cần gọi đến hàm nào phụ thuộc vào đối số truyền vào khi gọi hàm.

# Một số hàm không thể nạp chồng trong C++

## 1. Hàm chỉ khác nhau kiểu trả về

> **Chú ý:** Không thể nạp chồng hàm nếu chúng chỉ khác nhau kiểu dữ liệu trả về.

i

:

```
int foo() {
	return 10;
}

double foo() {  // compiler error
	return 0.5;
}
```

Hai hàm trên giống nhau về tham số (không có), vì vậy trình biên dịch sẽ báo lỗi.

## 2. Tham số hàm kiểu typedef

Khai báo **typedef** chỉ là một bí danh (không phải kiểu dữ liệu mới), vì vậy chương trình bên dưới sẽ gặp lỗi:

:

```
typedef int myint;
void print(myint value)
{
	cout << value << '\n';
}
void print(int value) // compiler error
{
	cout << value << '\n';
}
```

## 3. Tham số hàm kiểu con trỏ * và mảng []

Tham số hàm con trỏ * và mảng [] là tương đương. Lúc này, khai báo mảng [] được chuyển đổi ngầm định thành một con trỏ.

:

```
int foo(int *x);
int foo(int x[]); // compiler error
```

Hai hàm trên giống nhau về tham số (int*), vì vậy trình biên dịch sẽ báo lỗi.

## 4. Nạp chồng hàm và từ khóa const

Trước tiên chúng ta hãy xem hai ví dụ sau. Chương trình 1 biên dịch thất bại, nhưng chương trình 2 biên dịch và chạy tốt.

**Ví dụ 1:**

:

```
// PROGRAM 1 (Fails in compilation)
#include<iostream>
using namespace std;

void fun(const int i)
{
	cout << "fun(const int) called ";
}

void fun(int i)
{
	cout << "fun(int ) called ";
}

int main()
{
	const int i = 10;
	fun(i);

	return 0;
}
```

**Ví dụ 2:**

:

```
// PROGRAM 2 (Compiles and runs fine)
#include<iostream>
using namespace std;

void fun(char *a)
{
	cout << "non-const fun() " << a;
}

void fun(const char *a)
{
	cout << "const fun() " << a;
}

int main()
{
	const char *ptr = "Howkteam";
	fun(ptr); // output: const fun () Howkteam

	system("pause");
	return 0;
}
```

C++ cho phép nạp chồng hàm với tham số là const **chỉ khi** tham số const là **tham chiếu** hoặc **con trỏ**. Đó là lý do tại sao chương trình 1 gặp lỗi biên dịch, nhưng chương trình 2 hoạt động.

Trong chương trình 1, tham số '**i**' được truyền theo giá trị, vì vậy '**i**' trong fun() là **bản sao** của 'i' trong main(). Do đó, fun() không thể sửa đổi '**i**' của hàm main(). Vì vậy, không quan trọng việc '**i**' được nhận dưới dạng tham số **const** hay tham số bình thường.

Khi tham số là tham chiếu hoặc con trỏ, chúng ta có thể sửa đổi giá trị được tham chiếu hoặc được trỏ tới, do đó chúng ta có thể có hai phiên bản của hàm, một phiên bản có thể sửa đổi giá trị được tham chiếu hoặc trỏ tới, một phiên bản không thể thay đổi.

> **Chú ý:** C++ cho phép nạp chồng hàm với tham số là const chỉ khi tham số const là tham chiếu hoặc con trỏ.

## Kết luận

Qua bài học này, bạn đã nắm được những kiến thức về Nạp chồng hàm trong C++ (Function overloading).

Trong bài tiếp theo, chúng ta sẽ cùng tìm hiểu về HÀM CÓ ĐỐI SỐ MẶC ĐỊNH TRONG C++ (Default parameters).

Cảm ơn các bạn đã theo dõi bài viết. Hãy để lại bình luận hoặc góp ý của mình để phát triển bài viết tốt hơn. Đừng quên "**Luyện tập – Thử thách – Không ngại khó**".