## Bài 56: Nạp chồng hàm trong C++ (Function overloading)

**Dẫn nhập**

Ở bài học trước, bạn đã nắm được những kiến thức về [HÀM NỘI TUYẾN TRONG C++ (Inline functions)](https://www.howkteam.vn/course/khoa-hoc-lap-trinh-c-can-ban/ham-noi-tuyen-trong-c-inline-functions-3894).

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)](https://www.howkteam.vn/course/khoa-hoc-lap-trinh-c-can-ban/co-ban-ve-ham-va-gia-tri-tra-ve-basic-of-functions-and-return-values-1339)

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:

* [**C#**](https://howkteam.vn/learn)

**1 int** add(**int** a, **int** b)

2 {

3 **return** a + b;

4 }

5

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:

* [**C#**](https://howkteam.vn/learn)

**1 int** addInteger(**int** a, **int** b)

2 {

3 **return** a + b;

4 }

5

**6 double** addDouble(**double** a, **double** b)

7 {

8 **return** a + b;

9 }

10

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:

* [**C#**](https://howkteam.vn/learn)

**1 int** add(**int** a, **int** b)

2 {

3 **return** a + b;

4 }

5

**6 double** add(**double** a, **double** b)

7 {

8 **return** a + b;

9 }

10

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

* [**C#**](https://howkteam.vn/learn)

**1 int** add(**int** a, **int** b, **int** c)

2 {

3 **return** a + b + c;

4 }

5

**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.

* [**C#**](https://howkteam.vn/learn)

**1 int** foo() {

2 **return** 10;

3 }

4

**5 double** foo() { // compiler error

6 **return** 0.5;

7 }

8

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:

* [**C#**](https://howkteam.vn/learn)

1 typedef **int** myint;

**2 void** print(myint **value**)

3 {

4 cout << **value** << '\n';

5 }

**6 void** print(**int** **value**) // compiler error

7 {

8 cout << **value** << '\n';

9 }

**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.

* [**C#**](https://howkteam.vn/learn)

**1 int** foo(**int** \*x);

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

3

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:**

* [**C++**](https://howkteam.vn/learn)

1 // PROGRAM 1 (Fails in compilation)

2 #include<iostream>

**3 using** **namespace** **std**;

4

**5 void** fun(**const** **int** i)

6 {

7 **cout** << "fun(const int) called ";

8 }

9

**10 void** fun(**int** i)

11 {

12 **cout** << "fun(int ) called ";

13 }

14

**15 int** main()

16 {

17 **const** **int** i = 10;

18 fun(i);

19

20 **return** 0;

21 }

**Ví dụ 2:**

* [**C++**](https://howkteam.vn/learn)

1 // PROGRAM 2 (Compiles and runs fine)

2 #include<iostream>

**3 using** **namespace** **std**;

4

**5 void** fun(**char** \*a)

6 {

7 **cout** << "non-const fun() " << a;

8 }

9

**10 void** fun(**const** **char** \*a)

11 {

12 **cout** << "const fun() " << a;

13 }

14

**15 int** main()

16 {

17 **const** **char** \*ptr = "Howkteam";

18 fun(ptr); // output: const fun () Howkteam

19

20 system("pause");

21 **return** 0;

22 }

23

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).

“**Luyện tập – Thử thách – Không ngại khó**”.