

**C++**

**KHÔNG KHÓ**

# Giới thiệu:

C++ hiện nay là một ngôn ngữ phổ biến cho những người mới học lập trình.

Cuốn sách này được viết ra (~~cho vui~~) là những kiến thức mình đã được học trong lúc học C++. Mình mong cuốn sách này có thể cung cấp đủ kiến thức đến cho các bạn :Đ. Và nếu có gì sai sót thì nhớ báo lại cho mình nhé:3

Mình khuyên là trước khi bạn học, bạn nên có một số nền tảng về lập trình. Bởi vì có một số từ ngữ, kiến thức mà khi mình viết sách chưa được chuẩn lắm.

Các bạn còn có thể tìm kiếm các tài liệu xung quanh như khóa học của Howkteam, Codelearn.io, W3school,... rất là nhiều! Nhưng nếu bạn là người thích trải nghiệm những thứ mới mẻ thì bạn có thể đọc cuốn sách của mình :D

Cuốn sách này mình dựa trên kiến thức mình đã đọc ở trang w3school, các bạn có thể học tại đây (tiếng anh)

[C++ Tutorial \(w3schools.com\)](https://www.w3schools.com/C++Tutorial/)

# Mục lục: gồm 13 bài

- Bài 1: Hello world
- Bài 2: Đầu ra
- Bài 3: Ghi chú
- Bài 4: Biến
- Bài 5: Nhập dữ liệu
- Bài 6: Các kiểu dữ liệu
- Bài 7: Toán tử
- Bài 8: Xâu
- Bài 9: Toán trong C++
- Bài 10: Câu lệnh điều kiện If-Else
- Bài 11: Lặp
- Bài 12: Mảng
- Bài 13: Hàm

Tác giả Nguyễn Minh Hiếu (hieuhfgr)

Github: [hieuhfgr \(Hieu, Nguyen Minh\) · GitHub](#)

Bài giải: [hieuhfgr/cpp-khong-kho:  
sách C++ Không khó kèm lời giải bài  
tập \(github.com\)](#)

# Bài 1. Hello world

## 1.1 Get Started (bắt đầu)

Bạn cần hai thứ để bắt đầu học C++

1. Một Text Editor để viết chương trình C++. Ở đây mình sẽ sử dụng IDE là Visual Studio Code (các bạn có thể sài IDE khác như Code::Blocks, ...)
2. Một Compiler (trình biên dịch) để dịch ngôn ngữ C++ qua ngôn ngữ máy để máy tính có thể hiểu được.

## 1.2 Quickstart

1.2.1 Setup Visual Studio Code cho C++  
(google chứ mình lười ghi quá:>)

1.2.2 Chương trình đầu tiên (first Program)

Bây giờ bạn mở Visual Studio Code lên.

Sau đó chọn mục File -> New File

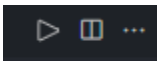
Sau đó vào File -> Save tên file là helloworld.cpp ở thư mục bạn muốn

Bây giờ, bạn hãy gõ chương trình C++ theo code bên dưới:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "hello world";
    return 0;
}
```

(Nếu bạn đang lo lắng có nghĩa là gì thì bạn cứ làm theo đi tí mình sẽ giải thích:> )

Bây giờ bạn hãy bấm nút hình tam giác ở giao diện Visual Studio Code  (nhớ là phải cài extension Code Runner đấy nhé:Đ)

Nếu mà ở trong cửa sổ Terminal của Visual Studio Code hiện chữ hello world thì bạn đã thành công rồi đấy!

## 1.3 Cú pháp của một chương trình cơ bản

Bây giờ hãy nhìn lại file bạn đã code:

```
#include <iostream>
using namespace std;
int main()
{
    cout << "hello world";
    return 0;
}
```

Giải thích:

Dòng 1:

```
#include <iostream>
```

Từ khóa `#include` dùng để khai báo thư viện, và ở đây là thư viện `iostream`

Dòng 2:

```
using namespace std;
```

Có nghĩa là chúng ta có thể sử dụng tên cho các objects và biến trong thư viện chuẩn

*(bạn cũng không cần hiểu sâu về hai câu lệnh ở dòng 1 và 2, chỉ cần nhớ gần như mọi chương trình, cần có 2 lệnh đó:D)*

Dòng 3:

```
int main()
```

Khai báo hàm Main. Mỗi chương trình phải có hàm Main. Bất cứ dòng lệnh nào trong dấu {} sẽ được thực hiện.

Dòng 5:

```
cout << "hello world";
```

Là câu lệnh dùng để in ra màn hình một xâu kí tự, chữ số, kí tự. Ở đây lệnh Cout được dùng để in ra câu “hello world”

Dòng 6:

```
return 0;
```

là câu lệnh trả giá trị khi thực hiện xong hàm. Hàm main() sẽ trả một giá trị dù cho chương trình chạy có thành công hay không

- trả giá trị 0: mọi thứ chạy thành công
- trả giá trị khác 0: đã có gì đó đã chạy sai và chương trình dừng lại

## Bài 2: Đầu ra (OUTPUT)

### 2.1 Kiến thức

#### 2.1.1 In màn hình

Lệnh “cout” và toán tử “<<” dùng để in ra màn hình một giá trị hoặc một xâu.



Ví dụ:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "hello world";
    return 0;
}
```

Chương trình khi chạy sẽ in ra dòng chữ “hello world”.

Bạn có thể thêm bất cứ lệnh “cout” mà bạn muốn, nhưng bạn phải đảm bảo nó ở trong hàm.

## 2.1.2 Xuống dòng

Để viết xuống dòng mới, chèn kí tự “\n” hoặc xài “endl”. Ví dụ:

```
#include <iostream>
using namespace std;

int main()
{
    cout << "dong 1: Hieu dep trai" << endl;
    cout << "dong 2: chao cac ban\n";
    cout << "dong 3: Hieu la owner cua cuon sach:D";
    return 0;
}
```

Khi chạy thì chương trình sẽ in ra lần lượt là:

```
dong 1: Hieu dep trai
dong 2: chao cac ban
dong 3: Hieu la owner cua cuon sach:D
```

Kiến thức chỉ có vậy thui:Đ

## 2.2 Bài tập

Viết chương trình in ra tên và tuổi của bạn

## Bài 3: Ghi chú (Comments)

### 3.1 Kiến thức

Comments có thể được dùng để giải thích các đoạn code trong C++, và nó giúp cho người đọc dễ hiểu, người viết dễ thực hiện và đôi khi còn để Test Code bla bla...

Comments có thể ở một dòng và ở nhiều dòng

Comments 1 dòng:

```
//đây là comments 1 dòng
```

Comments nhiều dòng:

```
/*  
đây là  
comment nhiều dòng  
ahihi  
*/
```

Mọi Comments trong C++ thì biên dịch đều bỏ qua

## 3.2 Bài tập

Bài này không có bài tập:Đ

## Bài 4: Biến

### 4.1 kiến thức

#### 4.1.1 Khái niệm

Biến được dùng để lưu trữ giá trị. Khi thực hiện chương trình, giá trị của biến có thể bị thay đổi

Trong C++, sẽ có nhiều kiểu dữ liệu (với những từ khóa khác nhau) và bạn sẽ được học ở bài sau:>

#### 4.1.2 Khai báo

\* Cú pháp:

```
<kiểu dữ liệu> <tên biến> = <giá trị>;
```

Giải thích: “kiểu dữ liệu” là một trong những kiểu dữ liệu trong C++, “tên biến” là tên bạn đặt cho biến (như “x”

hoặc “ten”). Dấu “=” là một phép gán và “giá trị” sẽ là giá trị của biến

Ví dụ:

```
int x = 5;  
cout << x;
```

Khi chạy chương trình thì màn hình sẽ in ra số “5”

Bạn cũng có thể khai báo tên biến trước và gán giá trị cho biến sau. Ví dụ:

```
int x;  
x = 5;  
cout << x;
```

\* Một số kiểu dữ liệu khác:

```
int myNum = 5;           // Kiểu số nguyên  
double myFloatNum = 5.99; // Kiểu số thực  
char myLetter = 'D';     // Kiểu kí tự  
string myText = "Hello"; // Kiểu chuỗi kí tự (nhớ khai báo thư viện  
string)  
bool myBoolean = true;   // Kiểu Logic (true or false)
```

\* In giá trị của biến: dùng lệnh cout. Ví dụ:

```
int tuoi = 1;  
cout << "so tuoi cua toi la: " << tuoi << endl;
```

In ra màn hình là so tuoi cua toi la: 1

\* Thêm các giá trị của biến lại với nhau: dùng toán tử “+”.

Ví dụ:

```
int x = 5;  
int y = 6;  
int sum = x + y;  
cout << sum;
```

\* Khai báo nhiều biến: bạn có thể khai báo nhiều biến có cùng kiểu dữ liệu. Ví dụ:

```
int x = 5, y = 6, z = 50;  
cout << x + y + z;
```

*\* nghe giảng hồ đồn cách này giúp bạn khai báo nhanh hơn :))) \**

### 4.1.3 Hằng và cách khai báo

Hằng dùng để lưu trữ giá trị, nhưng khác với biến, giá trị của hằng sẽ không bị thay đổi trong khi thực hiện chương trình

\* Cú pháp:

```
const <kiểu dữ liệu> <tên hằng> = <giá trị>;
```

ví dụ:

```
const double so_pi = 3.14;  
cout << so_pi;
```

### 4.1.4 Định danh biến và Quy tắc khai báo Biến và Hằng

\* Mọi biến và hằng sẽ được định danh bằng một cái tên nhất định (tên này là do bạn đặt)

“Có một tip từ giảng hồ là bạn nên đặt tên logic cho chương trình để sau này bạn có thể hiểu hoặc cho các đồng nghiệp của bạn hiểu”

\* Quy tắc đặt tên cho biết và hằng:

- Tên có thể chứa chữ số, chữ cái và dấu gạch dưới “\_”
- Tên phải bắt đầu bằng chữ cái hoặc dấu gạch dưới “\_”
- Tên phân biệt chữ hoa và chữ thường (ví dụ “TenCuaToi” và “tencuatoi” là các biến khác nhau)
- Tên không được chứa kí tự trắng hoặc các kí tự đặc biệt
- Tên không được trùng với các từ khóa

## 4.2 Bài tập

Bài 1: viết chương trình khai báo biến “a” có giá trị là 5 và in ra chương trình của “a”

Bài 2: viết chương trình khai báo hai giá trị của “x” và “y” lần lượt có giá trị là 4 và 5 là in ra chương trình tổng của hai biến đó

Bài 3: viết chương trình khai báo biến “tuoi” có giá trị là <tuổi của bạn> và in ra chương trình câu: “tuổi của tôi là {tuổi của bạn}”

## Bài 5: Nhập dữ liệu

### 5.1 Kiến thức

ở bài trước bạn đã học lệnh “cout” dùng để in giá trị ra màn hình. Bây giờ bạn sẽ học lệnh “cin” để lấy giá trị từ bàn phím

Lệnh “cin” kết hợp với toán tử “>>” sẽ gán giá trị nhập từ bàn phím cho cái biến. *\*nghe không hiểu đúng không? Vậy thì xem cú pháp nhé:Đ\** Cú pháp:

```
int x;  
cout << "x = ";  
cin >> x;  
cout << "gia tri ban vua nhap la: " << x;
```

ở ví dụ, khi user nhập số vừa nhập (giả sử là 5) và bấm phím “enter”, thì giá trị của x sẽ bằng số vừa nhập (số 5) và chương trình sẽ in ra là: “gia tri ban vua nhap la: 5”

## 5.2 Bài tập

Viết chương trình nhập một số từ bàn phím và in ra màn hình số vừa nhập cộng thêm 5

Ví dụ nhập “10” thì in ra màn hình là “15”

## Bài 6: Các kiểu dữ liệu

### 6.1 Kiến thức

ở trong bài biến, chúng ta đã học được mỗi biến sẽ có kiểu dữ liệu khác nhau. ở bài này, chúng ta sẽ tìm hiểu về các kiểu dữ liệu trong C++ nhé.

## 6.1.1 Kiểu dữ liệu nguyên thủy

Kiểu dữ liệu là kiểu dữ liệu được cung cấp sẵn trong các ngôn ngữ lập trình và được dùng để lưu trữ các giá trị đơn giản.

Bảng dưới đây cho ta biết về cụ thể từng loại trong các nhóm nguyên thủy *\*credit: viblo.asia\**

Kiểu	Mô tả	Mặc định	Kích thước	Ví dụ
boolean	true hoặc false	false	1 bit	true, false
byte	Số nguyên từ -128 .. 127	0	8 bits	123
char	Ký tự Unicode	\u0000	16 bits	'a', '\u0041', '\101', '\\', '"', '\n', '\b'
short	Số nguyên giá trị từ -32768 .. 32767	0	16 bits	1000
int	Số nguyên -2,147,483,648 .. 2,147,483,647	0	32 bits	-2, -1, 0, 1, 2



Kiểu	Mô tả	Mặc định	Kích thước	Ví dụ
long	Số nguyên dài	0	64 bits	-2L, -1L, 0L, 1L, 2L
float	Số thực	0.0	32 bits	1.23e100f, -1.23e-100f, .3f, 3.14F
double	Số thực	0.0	64 bits	1.23456e300d, -1.23456e-300d, 1e1d

Ngoài ra còn có thể sửa đổi bởi sử dụng một hoặc nhiều modifier dưới:

- signed (kiểu có dấu)
- unsigned (kiểu không có dấu)
- short
- long

xem thêm tại: *\*<https://quantrimang.com/kieu-du-lieu-trong-cplusplus-156173>\**

## 6.1.2 Kiểu dữ liệu số

Sử dụng kiểu `int` nếu bạn muốn lưu trữ số nguyên

Sử dụng kiểu `float` hoặc `double` nếu bạn muốn lưu số thập phân

Ví dụ:

+ int:

```
int so_nguyen = 123;  
cout << so_nguyen;
```

+ float:

```
float so_thuc_loai_float = 6.45;  
cout << so_thuc_loai_float;
```

+ double:

```
double so_thuc_loai_double = 456.46;  
cout << so_thuc_loai_double;
```

float so với double: *\*cre: w3schools\**

Độ **chính xác** của giá trị dấu phẩy động cho biết giá trị có thể có bao nhiêu chữ số sau dấu thập phân. Độ chính xác của **float** chỉ là sáu hoặc bảy chữ số thập phân, trong khi **double** các biến có độ chính xác khoảng 15 chữ số. Do đó sẽ an toàn hơn khi sử dụng **double** cho hầu hết các phép tính.

## 6.1.3 Kiểu Logic

Để lưu trữ dữ liệu **Logic** (True or False) thì chúng ta khai báo với kiểu dữ liệu và bool

Ví dụ:

```
bool isDepTrai = true;  
bool isHaveGirlFriend = false;  
cout << isDepTrai << endl;  
// Màn hình in ra là 1 (true)  
cout << isHaveGirlFriend << endl;  
// Màn hình in ra là 0 (false)
```

Biến sẽ chỉ lưu trữ hai giá trị duy nhất là **True** hoặc **False**.

Và khi giá trị được trả, **true = 1** và **false = 0**

## 6.1.4 Kiểu kí tự

Để lưu trữ kiểu kí tự, chúng ta sử dụng kiểu dữ liệu `char`.

Kí tự phải được bao quanh bởi dấu nháy đơn `'`.

Ví dụ:

```
char KiTuA = 'A';  
cout << KiTuA;
```

bạn chỉ có thể bỏ một kí tự ở trong dấu nháy đơn, nếu trong dấu nháy đơn từ 2 kí tự trở lên thì nó sẽ báo lỗi

```
char ten = 'Nguyen Van A';  
cout << ten;  
//khi chạy chương trình thì thông báo lỗi như sau:  
// warning: character constant too long for its type  
// char ten = 'Nguyen Van A';
```

Và bạn có thể sử dụng bảng mã ASCII.

```
char a;  
a = 65;  
cout << a;
```

*\*nhưng mà ai rảnh mà làm thế cho lằng nhằng :)))\**

## 6.1.5 Kiểu xâu kí tự

Khác với kiểu kí tự, xâu kí tự sẽ lưu trữ một xâu gồm nhiều kí tự ở trong xâu. Để lưu trữ một xâu kí tự, chúng ta xài kiểu dữ liệu `string`. Để xài, chúng ta phải thêm thư

viện `<string>` và đầu code. Và giá trị của string phải bao quanh dấu nháy kép “

*\*nói thế lằng nhằng lắm, coi ví dụ nè:>\**

```
string myName = "Nguyen Van A";  
cout << myName;
```

đơn giản nó là thế, còn nếu muốn học sâu thêm thì đợi bài sau:>

## 6.2 Bài tập

Bài 1. Viết chương trình khai báo biến có kiểu dữ liệu là số nguyên và gán giá trị là 7 và in ra màn hình

Bài 2. Viết chương trình tính diện tích hình vuông

Bài 3. Viết chương trình khai báo biến `myName` và gán giá trị là `<tên của bạn>` và in ra màn hình

## Bài 7: Toán tử

### 7.1 Kiến thức

Toán tử dùng để thực hiện các hoạt động trên biến và giá trị

#### 7.1.1 Toán tử số học

Toán tử số học dùng để tính toán các giá trị, biểu thức lại với nhau. (y như trong toán học)

*\*nói chung là không biết giải thích:)))\**

Phép toán	Miêu tả	Ví dụ
+	Phép cộng	$A + B$ kết quả là 30
-	Phép trừ	$A - B$ kết quả là -10
*	Phép nhân	$A * B$ kết quả là 200
/	Phép chia	$B / A$ kết quả là 2
%	Phép lấy số dư	$B \% A$ kết quả là 0
++	Toán tử tăng (++), tăng giá trị toán hạng thêm một đơn vị	$A++$ kết quả là 11
--	Toán tử giảm (--), giảm giá trị toán hạng đi một đơn vị	$A--$ kết quả là 9

## 7.1.2 Toán tử Gán

Toán tử gán dùng để đưa giá trị của một hằng số, biến số, một số biểu thức hoặc kết quả của một hàm vào biến được gán.

Toán tử	Miêu tả	Ví dụ
=	Toán tử gán đơn giản. Gán giá trị toán hạng bên phải cho toán hạng trái.	$C = A + B$ sẽ gán giá trị của $A + B$ vào trong C

+=	Thêm giá trị toán hạng phải tới toán hạng trái và gán giá trị đó cho toán hạng trái.	$C += A$ tương đương với $C = C + A$
-=	Trừ đi giá trị toán hạng phải từ toán hạng trái và gán giá trị này cho toán hạng trái.	$C -= A$ tương đương với $C = C - A$
*=	Nhân giá trị toán hạng phải với toán hạng trái và gán giá trị này cho toán hạng trái.	$C *= A$ tương đương với $C = C * A$
/=	Chia toán hạng trái cho toán hạng phải và gán giá trị này cho toán hạng trái.	$C /= A$ tương đương với $C = C / A$
%=	Lấy phần dư của phép chia toán hạng trái cho toán hạng phải và gán cho toán hạng trái.	$C \% = A$ tương đương với $C = C \% A$

## 7.1.3 Toán tử so sánh (hay là Toán tử quan hệ)

Toán tử so sánh dùng để so sánh hai giá trị với nhau.

	Miêu tả	Ví dụ
==	Kiểm tra hai giá trị có bằng nhau hay không	$(A == B)$ là không đúng
!=	Kiểm tra hai giá trị có khác nhau hay không	$(A != B)$ là true
>	Kiểm tra giá trị ở bên trái có lớn hơn giá trị của bên phải hay không. Nếu đúng thì trả về True	$(A > B)$ là không đúng
<	Kiểm tra giá trị ở bên trái có bé hơn giá trị của bên phải hay không. Nếu đúng thì trả về True	$(A < B)$ là true
>=	Kiểm tra giá trị ở bên trái có lớn hơn hoặc bằng giá trị ở bên phải hay không. Nếu đúng thì trả về True	$(A >= B)$ là không đúng
<=	Kiểm tra giá trị ở bên trái có bé hơn hoặc bằng giá trị ở bên phải hay không. Nếu đúng thì trả về True	$(A <= B)$ là true

## 7.1.4 Toán tử Logic

Toán tử Logic dùng để kiểm tra tính đúng đắn của một hoặc nhiều biểu thức

	Miêu tả	Ví dụ
&&	Được gọi là toán tử logic AND (và). Nếu cả hai toán tử đều có giá trị khác 0 thì điều kiện trở lên true.	(A && B) là false.
	Được gọi là toán tử logic OR (hoặc). Nếu một trong hai toán tử khác 0, thì điều kiện là true.	(A    B) là true.
!	Được gọi là toán tử NOT (phủ định). Sử dụng để đảo ngược lại trạng thái logic của toán hạng đó. Nếu điều kiện toán hạng là true thì phủ định nó sẽ là false.	!(A && B) là true.

## 7.2 Bài tập

Kiến thức này rất quan trọng nên không có bài tập. Bài này sẽ được áp dụng cho rất nhiều bài sau nên học cho kĩ vô: /

## Bài 8: Xâu

### 8.1 Kiến thức

Biến có kiểu dữ liệu là xâu sẽ lưu trữ xâu.

Ví dụ:

```
#include <iostream>
//thêm thư viện <string> để có thể xài kiểu dữ liệu string
#include <string>
using namespace std;

int main()
{
    string myCrushName = "Secret";
}
```

```
cout << "My Crush name is " << myCrushName;
return 0;
}
```

Khi chạy chương trình sẽ in ra đoạn chữ là `My Crush Name is Secret`, chúng ta sẽ thấy biến `MyCrushName` lưu trữ giá trị là xâu `"Secret"` giống như kiến thức đã học ở bài trước, Còn bài này chúng ta sẽ đi sâu hơn về Xâu.

## 8.1.1 Cộng Xâu

Sử dụng toán tử (phép toán) `+` để gộp 2 xâu lại với nhau để tạo ra một xâu mới từ hai xâu đã gộp.

Ví dụ:

```
string ten = "Hieu";
string ho = "Nguyen";
string HoVaTen = ho + ten;
cout << HoVaTen;
```

Khi chạy chương trình sẽ in ra dòng chữ `"NguyenHieu"`. Ở đây, bạn có thể thêm khoảng trắng vào xâu để có thể đẹp hơn.

```
string ten = "Hieu";
string ho = "Nguyen";
string HoVaTen = ho + " " + ten;
cout << HoVaTen;
```

Bạn có thể xài hàm `append()` để có thể cộng xâu. Ví dụ

```
string ten = "Hieu";
string ho = "Nguyen ";
string HoVaTen = ho.append(ten);
cout << HoVaTen;
// chương trình sẽ in ra màn hình là: Nguyen Hieu
```



*\*bạn có thể xài `append` hoặc toán tử `+` để cộng xâu nhưng mà không nên xài cả hai bởi vì có thể làm rối nào bạn:/\**

Không biết có ai như mình không khi mình đặt ra câu hỏi là liệu string có cộng được với number không nhỉ? Vì thế mình đã code như thế này:

```
#include <iostream>
//thêm thư viện <string> để có thể xài kiểu dữ liệu string
#include <string>
using namespace std;

int main()
{
    int a = 6;
    string b = "10";
    cout << b + a;
    return 0;
}
```

Và khi mình chạy chương trình thì THẬT BẤT NGỜ khi chương trình báo lỗi :/, vì thế bạn không thể cộng string với một number khi xài toán tử `+` được nhé ://

Nên nhớ:

`<Number> + <number>` thì cộng 2 giá trị

`<String> + <string>` thì ghép xâu

## 8.1.2 Độ dài xâu

Để mà có được độ dài của xâu, chúng ta xài hàm `length()`

Ví dụ:

```
string myName = "Hieudeptrai";  
int DoDaiXau = myName.length();  
cout << "do dai cua xau myName la: " << DoDaiXau;
```

Bởi vì chuỗi `myName` có 11 kí tự nên chương trình sẽ in ra là: `do dai cua xau myName la: 11`.

Bạn cũng có thể lấy độ dài của chuỗi bằng hàm `size()`. Cả hai hàm `size` và `length` đều lấy độ dài của chuỗi nên bạn có thể lựa chọn 1 trong 2 để sử dụng :D

```
string myName = "Hieudeptrai";  
cout << "Su dung ham Length, xau co do dai la: "<<myName.length() << endl;  
cout << "su dung ham Size, xau co do dai la: "<<myName.size();
```

### 8.1.3 Truy cập chuỗi

Để có thể truy cập đến các kí tự có trong chuỗi, bạn chỉ cần thêm dấu `[]` và bỏ thêm một số `x` trong dấu ngoặc vuông ( điều kiện  $0 \leq x < \text{<độ dài của chuỗi>}$  )

Nghe khá là khó hiểu nên mình sẽ lấy ví dụ nhé :c

```
string myName = "Hieudeptrai";  
cout << myName[0];  
//chương trình sẽ in ra là "H"
```

ở dòng thứ 2, mình đã thêm `index` là 0, vì thế chương trình sẽ dò kí tự đầu tiên của chuỗi `myName` là `H` và in ra màn hình, `index` sẽ bắt đầu từ 0, và `index` này phải nhỏ hơn độ dài của chuỗi.

Để mà đổi các kí tự có trong xâu thì mình sẽ truy cập đến vị trí của kí tự trong xâu cần thay đổi và gán giá trị mới. Ví dụ:

```
string myName = "Hieudeptrai";  
myName[0] = 'D';  
cout << myName;
```

lưu ý nhỏ, khi gán giá trị mới cho kí tự đã truy cập thì giá trị của kí tự mới phải bắt buộc đặt trong dấu nháy đơn, kí tự bao quanh dấu nháy kép thì chương trình sẽ báo lỗi.

## 8.1.4 Nhập xâu từ bàn phím

Bạn có thể dùng lệnh cin để nhập xâu từ bàn phím

```
string yourName;  
cout << "Nhap ten cua ban: ";  
cin >> yourName; //giả sử nhập là "hieu"  
cout << "Ten cua ban la: " << yourName;  
//chương trình in ra dòng chữ là "Te cua ban la: Hieu"
```

Nhưng nếu xâu bạn nhập từ bàn phím có chứa dấu cách thì lệnh cin sẽ lấy cái từ ngữ đầu tiên (kể cả bạn có gõ bao nhiêu từ đi nữa)

```
string yourName;  
cout << "Nhap ho va ten cua ban: ";  
cin >> yourName;  
cout << "Ten cua ban la: " << yourName;
```

chương trình sẽ in ra:

```
Nhap ho va ten cua ban: Hieu dep trai  
Ten cua ban la: Hieu
```

Để khắc phục tình trạng đó thì chúng ta sẽ dùng lệnh `getline()` với tham số thứ nhất là `cin` và tham số thứ hai là biến có kiểu dữ liệu là `string`

```
string yourName;  
cout << "Nhap ho va ten cua ban: ";  
getline(cin, yourName);  
cout << "Ten cua ban la: " << yourName;
```

tình trạng đã được khắc phục :D

```
Nhap ho va ten cua ban: Hieu dep trai  
Ten cua ban la: Hieu dep trai
```

## 8.2 Bài tập

Bài 1: viết chương trình khai báo một xâu và gán giá trị là tên của bạn

Bài 2: viết chương trình khai báo 2 biến họ và tên và thực hiện phép cộng xâu và in ra màn hình họ và tên

Bài 3: viết chương trình nhập một xâu bất kì từ bàn phím, sau đó in ra màn hình độ dài của xâu và in ra kí tự thứ 5 ở trong xâu

## Bài 9: Toán trong C++

Ở bài này sẽ làm quen với thư viện `cmath`, ai có hứng thú thì có thể vào trang web dưới và tự tìm hiểu

[https://www.w3schools.com/cpp/cpp\\_math.asp](https://www.w3schools.com/cpp/cpp_math.asp)

# Bài 10: Câu lệnh điều kiện If-Else

## 10.1 Kiến thức

Nhớ kiến thức của bài 7 hong? Đọc lại và chuẩn bị học nhé.

### 10.1.1 Câu lệnh điều kiện if (dạng thiếu)

Cú pháp:

```
if (điều kiện)
{
    câu lệnh
}
```

Nếu điều kiện đúng thì mọi câu lệnh nằm trong dấu `{}` sẽ được thực hiện

Ví dụ:

```
bool isDepTrai = true;
if (isDepTrai == true)
{
    cout << "Toi Dep trai";
};
```

Biến `isDepTrai` được gán giá trị là `true` và khi kiểm tra điều kiện thì thỏa mãn nên đã thực hiện câu lệnh trong dấu `{}` là in ra màn hình chữ `Toi Dep Trai`

Ví dụ rõ hơn:

```
#include <iostream>
using namespace std;

int main()
{
    int a = 1;
    int b = 3;

    if (a > b)
    {
        cout << "a lon hon b";
    };
    if (a < b)
    {
        cout << "a be hon b";
    };
    if (a == b)
    {
        cout << "a bang b";
    };
    return 0;
}
```

Chương trình này sẽ so sánh giá trị của hai biến a và b. Ở trong chương trình thì a = 1 và b = 3 nên câu lệnh điều kiện thứ 2 đúng.

## 10.1.2 Câu lệnh điều kiện if else (dạng đủ)

Cú pháp:

```
if (điều kiện) {
    // điều kiện đúng thì các câu lệnh trong đây sẽ thực hiện
```

```
} else {  
    // điều kiện sai thì các câu lệnh trong đây sẽ thực hiện  
}
```

Ví dụ:

Chương trình kiểm tra số âm hoặc dương

```
int a;  
cout << "A = "; cin >> a;  
if (a >= 0) {  
    cout << a << " là số dương";  
} else {  
    cout << a << " là số âm";  
}
```

ở trong ví dụ trên, khi ta nhập một số lớn hơn 0 thì chương trình kiểm tra điều kiện ra đúng nên sẽ thực hiện khối câu lệnh if, còn khi ta nhập một số bé hơn 0 thì điều kiện sai nên sẽ thực hiện khối câu lệnh ở phần else

## 10.1.3 Câu lệnh điều kiện if else if

Cú pháp:

```
if (điều kiện 1) {  
    // khối câu lệnh sẽ được thực hiện nếu điều kiện 1 đúng  
} else if (điều kiện 2) {  
    // khối câu lệnh sẽ được thực hiện nếu điều kiện 1 sai và điều kiện 2  
    đúng  
} else {  
    // khối câu lệnh sẽ thực hiện nếu cả điều kiện 1 và điều kiện 2 đều  
    sai  
}
```

Khó hiểu hả? Cho ví dụ nè:

```
int myAge = 20 ;  
if (myAge < 13) {  
    cout << "Bạn là một đứa con nít!";  
}
```

```
} else if (myAge < 18) {  
    cout << "Ban la mot dua thanh nien!";  
} else {  
    cout << "Ban la nguoi truong thanh!";  
}
```

Trong ví dụ, chúng ta thấy cấu trúc câu lệnh điều kiện if else if. Biến myAge được gán giá trị là 13 nên khi xét câu lệnh điều kiện đầu tiên (myAge < 13) thì điều kiện trả về false nên chương trình sẽ xét đến điều kiện tiếp theo.

Chạy đến điều kiện tiếp theo thì điều kiện (myAge < 18) trả về false nên chương trình đã thực hiện khối câu lệnh ở dòng else ở cuối (output: Ban la nguoi truong thanh!)

## 10.1.4 ngoài lề

Có một cách xài câu lệnh if else gọn hơn (còn gọi là phép toán 3 ngôi) thì nó có cú pháp như sau:

```
<biến> = (điều kiện) ? <giá trị đúng> : <giá trị sai>;
```

Ví dụ:

```
bool isDepTrai = false;  
string result = (isDepTrai == true) ? "Dep trai" : "Khong Dep trai";  
cout << result;  
//output: Khong Dep trai
```

Tạo biến result để chứa kết quả, khi gán thì chương trình sẽ kiểm tra điều kiện isDepTrai, mà biến isDepTrai được gán giá trị là false nên chương trình đã gán cho biến result là xâu “khong Dep trai”



*\*nếu bạn thấy phần ngoài lề không hiểu thì đừng quá quan tâm bởi vì bạn có thể xài cách nguyên thủy thay vì cách này:/\**

## 10.1.5 Switch Case

Trong một số trường hợp, thay vì xài câu lệnh if thì bạn có thể xài Switch case để dễ quản lí code hơn

Cú pháp:

```
switch(x) {  
  case <giá trị 1>:  
    // khối lệnh được thực hiện nếu x có giá trị 1  
    break;  
  case <giá trị 2>:  
    // khối lệnh được thực hiện nếu x có giá trị 2  
    break;  
  ...  
  case <giá trị n>:  
    // khối lệnh được thực hiện nếu x có giá trị là n  
    break;  
  default:  
    // khối lệnh được thực thi nếu x không bằng các giá trị ở trên  
}
```

Không hiểu ư? Cho ví dụ nè :D

```
int day = 4;  
switch (day) {  
  case 2:  
    cout << "Monday";  
    break;  
  case 3:  
    cout << "Tuesday";  
    break;  
  case 4:  
    cout << "Wednesday";  
    break;  
}
```

```
case 5:
    cout << "Thursday";
    break;
case 6:
    cout << "Friday";
    break;
case 7:
    cout << "Saturday";
    break;
case 1:
    cout << "Sunday";
    break;
}
// Outputs "Wednesday" (day 4)
```

Cách thức hoạt động:

Mọi `case` ở trong `switch` sẽ được so sánh với biến, nếu mà giá trị của `case` nào bằng giá trị của biến thì sẽ thực hiện khối câu lệnh ở dòng `case` đó, như ở trong ví dụ thì khi chạy đến `case 4` (case với giá trị là 4) sẽ bằng với biến `day` nên chương trình sẽ in ra là `"Wednesday"`. Còn từ khóa `Break` thì khi giá trị của `case` đó và chạy xong hết lệnh rồi thì `Break` giúp cho thoát ra khỏi `switch` (nếu không có từ khóa `Break` thì chương trình sẽ tiếp tục xét đến những `case` tiếp theo).

## 10.2 Bài tập

Bài 1: viết chương trình kiểm tra kiểm tra biến tuổi (biến này được nhập từ bàn phím) và kiểm tra xem có được lái xe máy hay không?

Bài 2: viết chương trình kiểm tra biến có chia hết cho 2 hay không (giá trị của biến được nhập từ bàn phím)

Bài 3: viết chương trình nhập 2 số từ bàn phím sau đó in ra số lớn hơn

## Bài 11: Lặp

### 11.1 Kiến thức

Vòng lặp có nghĩa là chương trình thực hiện nhiều lần khối câu lệnh trong vòng lặp

Vòng lặp giúp code của bạn không bị rối, giảm thời gian code và có thể giúp người đọc dễ hiểu hơn

#### 11.1.1 Vòng lặp For

Vòng lặp for được dùng để lặp khối lệnh trong vòng lặp với số lần biết trước

Cú pháp:

```
for ( init; condition; increment )  
{  
    //câu Lệnh muốn Lặp  
}
```

Init: thực hiện một lần và trước khi xử lí khối lệnh

Condition: điều kiện để câu lệnh trong vòng lặp được chạy (nếu điều kiện trả true thì sẽ thực hiện vòng lặp cho đến khi điều kiện trả false)

increment: dùng để tăng số đếm trong vòng lặp

ví dụ cụ thể:

```
cout << "bat dau chay vong lap:" << endl;
for (int i = 1; i < 11; i++)
{
    cout << "i = " << i << endl;
}
```

Biến `i` gán giá trị là `1` và được thực hiện các câu lệnh lần đầu tiên, sau đó kiểm tra điều kiện `i < 11` thì trả về `true` nên các câu lệnh trong vòng lặp vẫn tiếp tục thực hiện, khi thực hiện xong thì giá trị của `i` sẽ được tăng thêm một đơn vị (`i++`), sau một lúc thì giá trị của `i` lúc này là `11` và khi kiểm tra điều kiện thì trả về `false` nên kết thúc vòng lặp

Output:

```
bat dau chay vong lap:  
i = 1  
i = 2  
i = 3  
i = 4  
i = 5  
i = 6  
i = 7  
i = 8  
i = 9  
i = 10
```

## 11.1.2 Vòng lặp while

Vòng lặp while được dùng để lặp khối lệnh trong vòng lặp với số lần chưa biết trước

Cú pháp:

```
while (điều kiện) {  
    // các câu lệnh  
}
```

Ví dụ

```
int i = 1;  
while (i < 10) {  
    cout << "i = " << i << endl;  
    i++;  
}
```

Biến đếm `i` sẽ được gán giá trị là `1`, sau đó chương trình sẽ kiểm tra điều kiện `i < 10`, điều kiện đúng thì chương

trình sẽ thực hiện khối câu lệnh ở trong vòng lặp `while`, cho đến khi `i > 10` thì mới dừng lại

*\*đừng quên tăng hoặc giảm giá trị của biến đếm để tránh việc vòng lặp sẽ lặp không bao giờ dừng!\**

Ví dụ về việc vòng lặp không bao giờ dừng:

```
int i = 1;
while (i < 10) {
    cout << "i = " << i << endl;
}
```

Chương trình sẽ in ra mãi mà không được dừng!

## 11.1.3 Vòng lặp Do-While

Vòng lặp Do-while cũng giống như vòng lặp `while` *(nhưng khác ở chỗ là đổi vị trí của từ khóa Do và từ khóa While)* Nhưng khác ở chỗ là vòng lặp Do-while sẽ thực hiện khối câu lệnh trong vòng lặp 1 lần và mới bắt đầu kiểm tra điều kiện

Cú pháp:

```
do {
    // các câu lệnh
}
while (điều kiện);
```

ví dụ:

```
int i = 1;
do {
    cout << "i = " << i << endl;
    i++;
}
```

```
while (i < 10);
```

chương trình sẽ thực hiện câu lệnh in ra màn hình và gán giá trị, sau đó chương trình mới kiểm tra điều kiện (`i < 10`). Điều kiện đúng thì chương trình vẫn tiếp tục in ra màn hình cho đến khi `i` nhận giá trị là `10` thì chương trình mới dừng

## 11.1.4 Break và Continue

### 11.1.4.1 Break

Từ khóa `Break` dùng để “thoát ra” (kết thúc vòng lặp vòng lặp

Ví dụ:

```
int i = 1;
while (true) {
    cout << "i = " << i << endl;
    i++;
    //nếu i = 10 thì thoát vòng lặp
    if (i == 10){
        break;
    }
}
```

Khi chạy chương trình thì khi kiểm tra điều kiện thì LUÔN LUÔN đúng nên chương trình bắt đầu thực hiện các câu lệnh trong vòng lặp, sau đó kiểm tra điều kiện `if` trong vòng lặp (giải sử `i` đã được gán giá trị là `10`) nếu điều kiện đúng thì từ khóa `break` được thực hiện -> kết thúc vòng lặp While

## 11.1.4.1 Continue

Từ khóa `continue` dùng để thoát ra 1 lặp của vòng lặp (lưu ý vòng lặp sẽ không dừng hẳn mà chỉ skip 1 lần)

Ví dụ:

```
int i = 1;
while (i < 10) {
    //nếu i = 5 thì không in ra i = 5
    if (i == 5){
        i++;
        continue;
    }
    cout << "i = " << i << endl;
    i++;
}
```

Output:

```
i = 1
i = 2
i = 3
i = 4
i = 6
i = 7
i = 8
i = 9
```

## 11.2 Bài tập

Bài 1: Viết chương trình nhập một số n từ bàn phím sau đó in ra các số từ 1 -> n

Bài 2: Viết chương trình nhập một chuỗi ký tự từ bàn phím, sau đó bắt đầu in ra từng ký tự ở trong chuỗi vừa nhập



Bài 3: viết chương trình nhập số x từ bàn phím, sau đó kiểm tra số x có phải là số nguyên tố hay không

## Bài 12: Mảng

### 12.1 Kiến thức

Mảng dùng để lưu trữ nhiều biến trong một biến mảng.

Mảng sẽ tốt hơn cho việc lưu trữ thay vì xài biến.

#### 12.1.1 khai báo và gán giá trị cho mảng

\* khai báo:

```
<kiểu dữ liệu> <tên mảng>[<số phần tử trong mảng>];
```

ví dụ:

```
string students[20];
```

ví dụ trên bạn đã khai báo mảng students có 20 phần tử ở trong mảng.

bạn có thể gán giá trị luôn cho mảng ở phần khai báo (số phần tử có trong mảng không được lớn hơn số phần tử được khai báo)

```
string student[4] = {"VAN A", "VAN B", "VAN C", "VAN D"};
```

Bạn có thể không khai báo kích thước của mảng, lúc này số phần tử của mảng sẽ bằng số phần tử đã khởi tạo, như ví dụ dưới:

```
string students[] = {"VAN A", "VAN B", "VAN C", "VAN D" };  
//mảng có 4 phần tử
```

\* ví dụ về truy cập vào các phần tử của mảng

```
string students[4] = {"VAN A", "VAN B", "VAN C", "VAN D" };  
cout << students[0];  
// output: Van A
```

Phần tử đầu tiên trong mảng sẽ có index là 0, phần tử thứ 2 sẽ có index là 1, từ đó cộng lên:V

\* thay đổi giá trị của phần tử ở trong mảng:

Chúng ta sẽ truy cập đến phần tử cần thay đổi giá trị, sau đó sẽ gán giá trị mới cho phần tử, như ví dụ ở dưới.

```
string students[4] = {"VAN A", "VAN B", "VAN C", "VAN D" };  
cout << students[0] << endl;  
// output: Van A  
students[0] = "Van O";  
cout << students[0];  
// output: Van O
```

## 12.1.2 Sử dụng vòng lặp cho mảng

Việc sử dụng vòng lặp cho mảng sẽ giúp chúng ta xử lý các phần tử trong mảng dễ hơn, người viết dễ quản lý hơn,...

Ví dụ về việc sử dụng vòng lặp trong mảng:

```
string students[4] = {"VAN A", "VAN B", "VAN C", "VAN D" };  
for (int i = 0; i < 4; i++) {  
    cout << students[i] << endl;  
}
```

Chương trình sẽ thực hiện in ra màn hình lần lượt là giá trị của phần tử (ở ví dụ là tên của mảng students) bằng cách sử dụng vòng lặp for. Output:

```
VAN A  
VAN B  
VAN C  
VAN D
```

## 12.2 Bài tập

Bài 1: Viết chương trình tạo một mảng tên (giới hạn phần tử: 100) sau đó gán giá trị từng phần tử trong mảng tên từ bàn phím và in ra dữ liệu vừa nhập (nhập xong hết dữ liệu thì mới tiến hành in)

Bài 2: Tạo một mảng với kiểu số nguyên, gán giá trị từng phần tử trong mảng từ bàn phím, sau đó tiến hành kiểm tra các phần tử trong mảng chia hết cho 2 hay không? (Nếu chia hết thì in ra màn hình)

Bài 3: Tạo một mảng với kiểu số nguyên, gán giá trị từng phần tử trong mảng từ bàn phím, sau đó tiến hành xác

định xem có bao nhiêu phần tử trong mảng là số nguyên tố

## Bài 13: Hàm

### 13.1 Kiến thức

#### 13.1.1 Hàm là gì?

Hàm là một khối các lệnh mà chỉ được thực hiện khi gọi hàm đó ra. Các hàm sẽ thực hiện một chức năng nhất định và có thể tái sử dụng hàm (thực hiện hàm nhiều lần mà không phải code nhiều dòng)

#### 13.1.2 Tạo một hàm và cách gọi hàm

Cú pháp:

```
<kiểu dữ liệu> <tên hàm>() {  
    //khối lệnh  
}
```

Nhìn thì chắc bạn không hiểu đâu nên giờ mình sẽ cho ví dụ về một hàm in ra màn hình nhé.

```
#include <iostream>  
using namespace std;  
//tạo hàm in ra màn hình  
void inManHinh() {
```

```

    cout << "nguai viet cuon sach nay dep trai";
}

int main() {
    inManHinh(); //đến đây chương trình bắt đầu thực hiện các lệnh trong
hàm
    return 0;
}

```

Giải thích: các bạn sẽ thấy trong code có một hàm tên là `inManHinh`. Hàm đó được khai báo với kiểu dữ liệu là `Void` (`void` là kiểu dữ liệu không cần hàm trả về một giá trị). ở trong dấu `{}` của hàm `inManHinh` có một lệnh `cout` sẽ in ra màn hình là “nguai viet cuon sach nay dep trai”. Và khi chạy, chương trình sẽ bắt đầu thực hiện khối lệnh ở trong hàm (`inManHinh`)

\* cách gọi hàm: như ví dụ trên, trong `main` sẽ có tên hàm (`inManHinh`) và khi chương trình dịch đến đây thì nó đã gọi hàm (đồng nghĩa với việc các lệnh ở trong hàm sẽ được thực hiện)

Một hàm có thể được gọi nhiều lần (ví dụ ở dưới)

```

#include <iostream>
using namespace std;

void inManHinh() {
    cout << "Toi dep trai" << endl;
}

int main() {
    //hàm được gọi 4 lần
    inManHinh();
    inManHinh();
}

```

```
    inManHinh();  
    inManHinh();  
    return 0;  
}
```

Output:

```
Toi dep trai  
Toi dep trai  
Toi dep trai  
Toi dep trai
```

*Dễ hiểu khum?:>*

Lưu ý, bạn không được để phần khai báo hàm ở sau phần `main` của chương trình!

```
int main() {  
    //hàm được gọi 4 lần  
    inManHinh();  
    inManHinh();  
    inManHinh();  
    inManHinh();  
    return 0;  
}  
  
void inManHinh() {  
    cout << "Toi dep trai" << endl;  
}
```

Khi chạy thì chương trình sẽ báo lỗi

Nhưng mà bạn có thể khai báo hàm trước phần main và sau đó bắt đầu bỏ lệnh vô trong hàm (như ở dưới):

```
#include <iostream>  
  
using namespace std;  
  
void inManHinh(); //khai báo hàm
```

```

int main() {
    //hàm được gọi 4 lần
    inManHinh();
    inManHinh();
    inManHinh();
    inManHinh();
    return 0;
}

// sau đó bỏ lệnh vô trong hàm
void inManHinh() {
    cout << "Toi dep trai" << endl;
}

```

### 13.1.3 Hàm có tham số

*\*Nhìn cái tên khó hiểu thôi chứ nó dễ hiểu lắm:)))\**

Tham số là gì? Tham số là các biến được sử dụng trong một hàm mà giá trị của biến đó được cung cấp từ lúc gọi hàm, các tham số này ngăn cách bằng dấu “,” và có cú pháp giống với khai báo biến.

Cú pháp:

```

<kiểu_dữ_liệu> tenHam(thamso1, thamso2, thamso3, thamson) {
    // khối lệnh
}

```

Ví dụ:

```

void GoiTenToi(string ten) {
    cout << ten << " muon duoc goi ten";
}

int main() {
    GoiTenToi("Bread");
}

```

```
    return 0;
}
```

ở đây, hàm `GoiTenToi` có chức năng in ra màn hình tên khi được truyền tham số (tham số được truyền là `"Bread"`)

\* đặt tham số mặc định

Khi khai báo tham số, bạn có thể gán luôn giá trị mặc định cho tham số. và khi gọi hàm, nếu hàm không được truyền tham số thì chương trình sẽ lấy cái tham số mặc định mà bạn đã khai báo từ trước còn không thì chương trình sẽ lấy tham số đã truyền. ví dụ bên dưới

```
void GoiTenToi(string ten = "super idol") {
    cout << ten << " muon duoc goi ten" << endl;
}

int main() {
    GoiTenToi("bread"); // in ra la "bread muon duoc goi ten"
    GoiTenToi(); // in ra la "super idol muon duoc goi ten"
    return 0;
}
```

ở trong một hàm, bạn có thể thêm bất cứ tham số gì bạn muốn (tất nhiên là thêm tham số có chủ đích)

ví dụ:

```
void inManHinh(string name, int age) {
    cout << "Ten: " << name << ". ";
    cout << "tuoi: " << age << ", " << endl ;
}

int main() {
```



```
inManHinh("Liam", 3);  
inManHinh("Jenny", 14);  
inManHinh("Anja", 30);  
return 0;  
}
```

Tham số của hàm inManHinh có lần lượt là `name` và `age`.  
Nên nhớ, khi gọi hàm thì bạn phải truyền đúng kiểu dữ liệu của tham số đã khai báo ở hàm.

### 13.1.4 Hàm trả giá trị

Trả giá trị ở đây là khi bạn gọi một hàm, sau khi xử lý xong, hàm sẽ trả cho bạn một giá trị nào đó (tùy theo chủ đích của bạn)

ở đây, bạn sẽ không sử dụng kiểu dữ liệu là void (sử dụng void thì hàm không trả giá trị), mà bạn sẽ sử dụng các kiểu dữ liệu như số nguyên, số thực, kí tự, xâu kí tự, kiểu logic,... và ở trong hàm sẽ có một từ khóa return. Từ khóa return này sẽ giúp trả giá trị

ví dụ:

```
int addTwoNumber(int a, int b) {  
    return a + b;  
}  
  
int main() {  
    cout << addTwoNumber(5,6);  
    return 0;  
}
```

Bạn có thể lưu trữ giá trị mà hàm trả về vào biến

```

int addTwoNumber(int a, int b) {
    return a + b;
}

int main() {
    int sum = addTwoNumber(5,6);
    cout << sum;
    return 0;
}

```

## 13.1.5 Chồng hàm

Thay vì bạn phải tạo nhiều hàm cho một mục đích với nhiều kiểu dữ liệu khác nhau, bạn có thể sử dụng phương pháp chồng hàm

Bên dưới là cách không sử dụng chồng hàm (khai báo nhiều hàm chỉ vì một mục đích cho nhiều kiểu dữ liệu)

```

int plusFuncInt(int x, int y) {
    return x + y;
}

double plusFuncDouble(double x, double y) {
    return x + y;
}

int main() {
    int myNum1 = plusFuncInt(8, 5);
    double myNum2 = plusFuncDouble(4.3, 6.26);
    cout << "Int: " << myNum1 << "\n";
    cout << "Double: " << myNum2;
    return 0;
}

```

Ở dưới đây là cách sử dụng chồng hàm

```

int plusFunc(int x, int y) {
    return x + y;
}

```

```
double plusFunc(double x, double y) {  
    return x + y;  
}  
  
int main() {  
    int myNum1 = plusFunc(8, 5);  
    double myNum2 = plusFunc(4.3, 6.26);  
    cout << "Int: " << myNum1 << "\n";  
    cout << "Double: " << myNum2;  
    return 0;  
}
```

## 13.2 Bài tập

Bài 1: viết hàm kiểm tra số nguyên tố

Bài 2: tính tổng các phần tử ở trong mảng bằng cách sử dụng hàm tính tổng hai số (giá trị phần tử được nhập từ bàn phím)

Bài 3: Viết hàm in ra “Ban that ngau”

# Tổng kết

Wow! Thật không thể tin là bạn đọc xong hết kiến thức đấy! Bạn thật ngẫu. Nhưng mà, đây chỉ mới là một phần kiến thức cơ bản của C++. Mình mong cuốn sách này sẽ giúp ích cho bạn trong con đường lập trình sắp tới nhé!

Kí tên:

Hieuhfgr

Nguyễn Minh Hiếu

17 - 02 - 2022