Chào mừng các bạn trở lại với Series Lập Trình Căn Bản C. Hôm nay chúng ta sẽ tiếp tục bài học về Hàm trong ngôn ngữ C.

Bài viết này yêu cầu các bạn có kiến thức ở những bài viết trước, những bạn nào chưa xem các bài viết trước thì có thể xem ở đây:

Bài 01: Chào mừng đến thế giới lập trình.

Bài 02: Kiểu dữ liệu trong ngôn ngữ C.

Bài 03:Nhập xuất trong ngôn ngữ C.

Bài 04:Toán tử trong ngôn ngữ C.

Bài 05:Ép kiểu trong ngôn ngữ C.

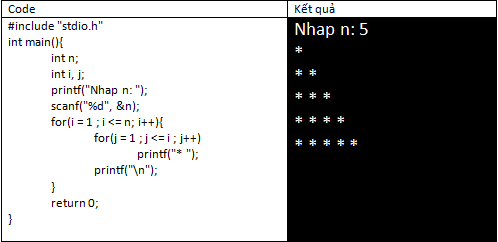
Bài 06:Câu lệnh rẽ nhánh trong ngôn ngữ C.

Hôm nay, chúng ta sẽ bắt đầu bài học về hàm với phần dẫn nhập.

**I, Giới thiệu về HÀM**

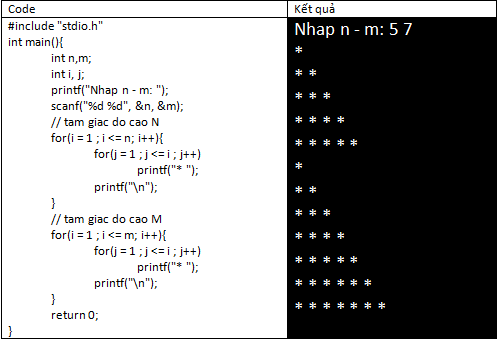
***Hàm là gì? Vì sao chúng lại xuất hiện?***

Giả sử bạn được thầy (cô) giáo đưa cho một bài toán, viết chương trình in ra một hình tam giác. Nó không quá khó và bạn có thể viết được, code như sau:

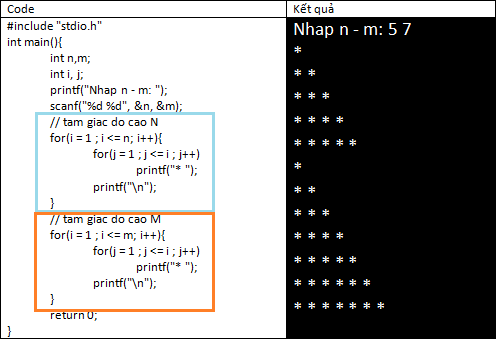


|  |  |
| --- | --- |
| Code | Kết quả |
| #include "stdio.h"  int main(){  int n;  int i, j;  printf("Nhap n: ");  scanf("%d", &n);  for(i = 1 ; i <= n; i++){  for(j = 1 ; j <= i ; j++)  printf("\* ");  printf("\n");  }  return 0;  } | Nhap n: 5  \*  \* \*  \* \* \*  \* \* \* \*  \* \* \* \* \* |

Vậy giả sử như chương trình muốn in 2 tam giác thì sao? Code chúng ta sẽ như thế này



Nhìn lại code của chúng ta, nó đã có sự trùng lặp về cú pháp ở đâu đây.



Hãy cùng tôi cải tiến đoạn code này, thay vì sử dụng 2 đoạn code riêng thì chúng ta sẽ thay đổi để có thể ngắn gọn hơn và linh động hơn.

|  |  |
| --- | --- |
| Code ban đầu | Cải tiến |
| #include "stdio.h"  int main(){  int n,m;  int i, j;  printf("Nhap n - m: ");  scanf("%d %d", &n, &m);  // tam giac do cao N  for(i = 1 ; i <= n; i++){  for(j = 1 ; j <= i ; j++)  printf("\* ");  printf("\n");  }  // tam giac do cao M  for(i = 1 ; i <= m; i++){  for(j = 1 ; j <= i ; j++)  printf("\* ");  printf("\n");  }  return 0;  } | #include "stdio.h"  int main(){  int n,m;  int i, j;  printf("Nhap n - m: ");  scanf("%d %d", &n, &m);  InTamGiac(n);  InTamGiac(m);  return 0;  } |

Wow, khá ngắn gọn ấy nhỉ.

So sánh 2 code, ta thấy code bên phải nhìn đẹp hơn phải không các bạn, và nó khiến chương trình dễ nhìn hơn. Đấy chính là **Hàm ( chương trình con)** mà chúng ta tìm kiếm bấy lâu.

**Hàm ( chương trình con ) là một đoạn code riêng biệt và có tên riêng.**

Trước khi đến với bài học, tôi sẽ cho các bạn thấy code đầy đủ của bài tập trên

#include "stdio.h"

void InTamGiac(int n){

int i, j;

for(i = 1 ; i <= n; i++){

for(j = 1 ; j <= i ; j++)

printf("\* ");

printf("\n");

}

}

int main(){

int n,m;

printf("Nhap n - m: ");

scanf("%d %d", &n, &m);

InTamGiac(n);

InTamGiac(m);

return 0;

}

Sau khi đã hiểu về hàm, chúng ta sẽ phân tích về cú pháp hàm:

**Cú pháp:**

**Kiểu\_trả\_về Tên\_hàm ( danh\_sách\_tham\_số <nếu có>)**

**{**

**// thân hàm.**

**[Return giá\_trị;]**

**}**

1. Kiểu trả về

Mình phân Kiểu\_trả\_về thành 2 loại. Có trả về và không trả về.

|  |  |  |  |
| --- | --- | --- | --- |
| **Tên kiểu** | **Đại diện** | **Trả về** | **Ghi chú** |
| Có trả về giá trị | int | Trả về kiểu nguyên | Sử dung **return** để trả về giá trị |
| float | Trả về kiểu thực |
| char | Trả về kiểu ký tự |
| Không trả về giá trị | void | Không trả về giá trị |

1. Tên Hàm: được đặt giống như tên biến.

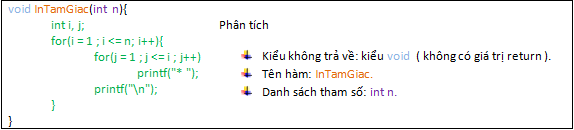
Ví dụ: tinhTong, Tinh\_Tong,….

Chú ý: Tên hàm không được giống với tên biến có trong hàm ( sẽ làm rõ hơn ở ví dụ).

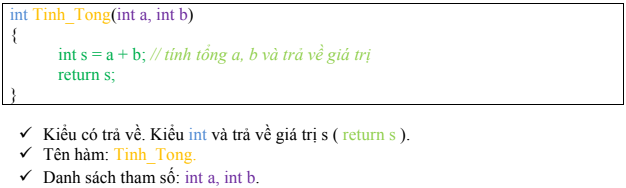
1. Danh sách tham số: Là những biến mà ta cần truyền vào và sử dụng trong hàm ( hãy xem ví dụ để hiểu rõ thêm).

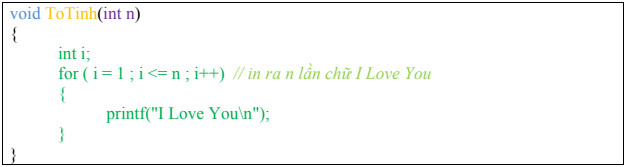
Mình sẽ ví dụ một hàm để các bạn hiểu rõ hơn về nó.

Ví dụ: Viết một hàm sử dụng để int ra tam giác.



Ví dụ 2: Viết một hàm tính tổng 2 số





Ở những ví dụ trên, các bạn đã biết cách khai báo về Hàm, nhưng cụ thể bên trong thì vẫn chưa biết, chúng ta hãy cùng đi tìm hiểu nhé.

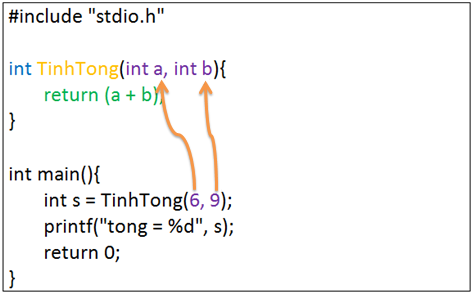
1. **Danh sách tham số.**

Danh sách tham số chính là những biến mà bạn sẽ truyền vào hàm để sử dụng nó.

Ví dụ như bạn muốn tính tổng 2 số, bạn cần truyền 2 số vào.

Bạn muốn tính diện tích hình chữ nhật, bạn truyền chiều dài và chiều rộng vào,…

Hình vẽ minh họa



Ở đoạn code này, mình đã truyền vào hàm 2 giá trị 6, 9. Ở hàm nhận 2 tham số **int a, int b**.

Và mình đã sử dụng 2 tham số đó để trả về kết quả là tổng của a, b.

**Lưu ý: Tham số truyền vào có thể có hoặc không**.

Như vậy là chúng ta đã biết về tham số được truyền vào, giờ đến với giá trị trả về

1. **Return**

Không biết các bạn thấy quen với nó không nhỉ, **return**.

Trong hàm main, các bạn đã thấy được cấu trúc của một hàm, cụ thể thế nào thì các bạn sẽ tự phân tích nhé, nhưng có một cái ở gần cuối là dòng **return 0;**

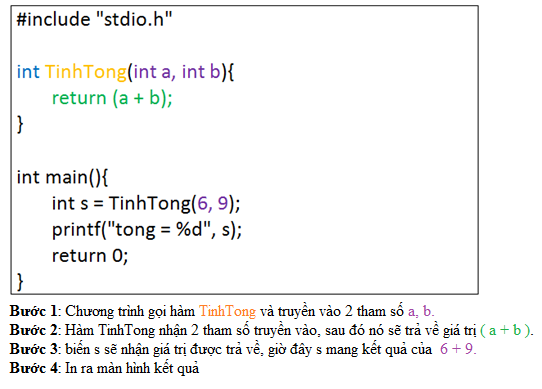
**Return là từ khóa trả về một cái gì đó cho ai đó.**

Ở hàm main có **return 0;** 🡪 Trả về cho hệ điều hành giá trị 0.

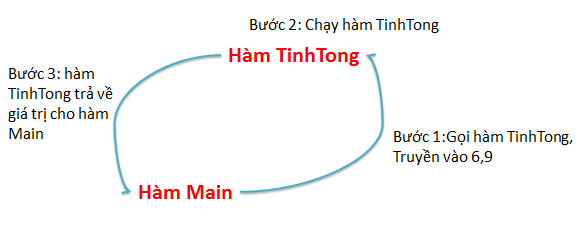
Ở hàm **TinhTong** có **return (a + b)**; 🡪 Trả về giá trị a + b cho biến s.

Tương tự như vậy, các bạn có thể suy luận ra thêm về từ khóa **return.**

Để có thể giúp bạn hiểu rõ hơn, tôi sẽ giải thích lại rõ ràng về chương trình trên.



Hay quá trình chạy được biểu diễn như hình dưới đây



Ở ví dụ trên tôi đã nói rõ hơn về cách chạy của Hàm, giờ chúng ta sẽ đến một số lưu ý nhỏ ở kiểu trả về:

* **Hàm có thể hoặc không có kiểu trả về.**
* **Kiểu trả về của hàm void là return; ( không có tham số trả về).**
* **Dữ liệu được return nên giống với kiểu trả về. Nếu khác thì kết quả có thể bị sai.**

Ví dụ chương trình trả về một số thực.

#include “stdio.h”

float TinhThuong(int a, int b)

{

return (float)(a/b);

}

int main()

{

int a = 10 , b = 3;

float s = TinhThuong(a,b);

printf(“s = %.2f”, s); // in ra 3.33

}

Đoạn code trên tôi đã sử dụng ép kiểu và trả về kiểu float, giả sử bỏ kiểu float ra hoặc thay đổi dữ liệu trả về là kiểu int thì sao? Các bạn chạy thử và lí giải nó nhé.

**II, Lời gọi hàm trong main và một số lưu ý.**

1. **Lời gọi hàm trong hàm Main**

Hàm có thể được đặt trên hoặc dưới hàm main.

Ta không xét khi đặt ở trên hàm main vì nó không ảnh hưởng nhiều đến chương trình.

NHƯNG nếu các bạn đặt hàm ở dưới hàm main thì phải khai báo prototype ( nguyên

mẫu hàm).

Và mình sẽ chỉ các bạn cách gọi hàm và khai báo prototype.

**a, Prototype.**

* Vị trí đặt: được đặt ở dưới lời gọi thư viện ( #include <stdio.h>) và trên hàm

main.

* Cú pháp: Tương tự cú pháp hàm. Các bạn copy y chang cú pháp hàm và thêm

dấu chấm phẩy “;”.

VD: hàm ToTinh ==> void ToTinh(int n);

Hàm tính tổng ==> int Tinh\_Tong(int a, int b);

Và đây là cách chính xác ta khai báo.

#include <stdio.h>

void ToTinh(int n);

int Tinh\_Tong(int a, int b);

int main()

{

// thân hàm

}

b, Gọi hàm trong main: Các bạn nhìn ví dụ dưới đây. Tôi sẽ đi kèm chú thích

trong code

#include <stdio.h>

void ToTinh(int n) // n nhận và thực thi lệnh trong hàm

{

int i;

for ( i = 1 ; i <= n ; i++)

{

printf("I Love You\n");

}

}

int main()

{

int n = 5;

ToTinh(n); // gọi hàm thì ta đưa biến vào. Ở đây là biến n

}

Hoặc ví dụ khác.

#include <stdio.h>

int Tinh\_Tong(int a, int b) // Hàm nhận giá trị a, b và sử dụng nó

{

int s = a + b; // thực hiện tính tổng

return s; // Trả về giá trị

}

/\*

hàm tính tổng còn có cách làm khác ngắn hơn là:

int Tinh\_Tong(int a, int b) // Hàm nhận giá trị a, b và sử dụng nó

{

return a+b; // Trả về luôn tổng a, b mà không cần phải sử dụng biến s.

}

\*/

int main()

{

int a = 7, b = 12;

int s = Tinh\_Tong(a,b); // đưa biến a, b vào hàm .

printf(“s = %d”,s); // Sau khi giá trị trả về thì nó sẽ đưa vào s. Ta in tổng.

}

1. Một số lưu ý
   1. Tính độc lập trong Hàm: Đây là một tính chất rất quan trọng với hàm.

Các bạn chú ý rằng. Hàm main và hàm các bạn tạo có thể cùng chứa một biến, nhưng giá trị có thể khác nhau. Xem ví dụ.

#include <stdio.h>

void TimX(int x)

{

x += 2;

printf("x trong = %d", x); // in ra 7

}

int main()

{

int x = 5;

TimX(x);

printf("x ngoai = %d", x); // in ra 5

}

Vì sao cùng một tên gọi, cùng một giá trị mà x ở trong hàm TimX và hàm main lại khác nhau?

Giải thích

Vì: Giá trị trong hàm TimX của chúng ta chỉ mới tính trong hàm, chưa trả về. ( ta sử dụng void)

Nếu thay đổi một chút. Không sử dụng void mà sử dụng int

#include <stdio.h>

int TimX(int x)

{

x += 2;

printf("x trong = %d", x); // in ra 7

}

int main()

{

int x = 5;

TimX(x);

printf("x ngoai = %d", x); // in ra 5

}

==> kết quả vẫn như vậy.

Nếu muốn thay đổi kết quả, ta phải có trả về giá trị, return x;

#include <stdio.h>

int TimX(int x)

{

x += 2;

printf("x trong = %d", x); // in ra 7

return x;

}

int main()

{

int x = 5;

x = TimX(x);

printf("x ngoai = %d", x); // in ra 7

}

Bây giờ mới thật sự trả về giá trị x.

Vậy khi làm một hàm bạn phải chú ý đến tính độc lập của hàm. Xét giá trị return.

Khi sang mảng các bạn sẽ được biết thêm một khái niệm mới. Mảng. Và mảng có cái hay là sử dụng void vẫn thay đổi được giá trị ở trong mảng. Hãy chờ xem.

Bài học về Hàm của mình đã kết thúc, giờ chúng ta cùng khởi động với một số bài tập nhé.

Bài 1: Viết hàm tính tổng từ 1 tới n

Bài 2: Viết hàm tính biểu thức sau:

Gợi ý: Đưa vào hàm giá trị n, tính tổng và trả về s. ( sử dụng kiểu float và nhớ ép kiểu).

Bài 3: Viết hàm tính mũ x. ( x là giá trị bạn nhập vào).

VD: int TinhMu(int x, int n)

sử dụng nó để tính mũ.

Bài 4: Viết hàm tính tổng 2 số nguyên, sử dụng nó để tính tổng 3 số.

Các bạn đã được làm quen với Số nguyên tố, số chính phương, số hoàn hảo... ở các bài tập trước. Và giờ các bạn hãy viết hàm cho những số đó.

Bài 5: Nhập vào một số và kiểm tra xem:

* Đó có phải là số hoàn hảo hay không.
* Đó có phải là số chính phương hay không.
* Đó có phải là số nguyên tố hay không.

Bài 6: Nhập vào một dãy số, kiểm tra xem đó có phải là số đối xứng hay không.

Số đối xứng là số mà mình đảo nó lại vẫn y như cũ:

Ví dụ: 6 là số đối xứng, 11 là số đối xứng, 151 là số đối xứng, 1551 cũng là số đối xứng trong khi số 12, 14, 1662 không phải là số đối xứng.