Bài 27. Hàm trong C

Bởi **Nguyễn Văn Hiếu** -

Bài số 25 trong 69 bài của khóa học Học C Không Khó

Trong bài học này, **Lập trình không khó** sẽ hướng dẫn các bạn các kiến thức căn bản nhất về **hàm trong C** (tên gọi khác là **chương trình con**). Mình sẽ đưa ra lý do tại sao nên dùng hàm, cách gọi hàm ở trong ngôn ngữ C và cách hoạt động của một chương trình có sử dụng các hàm con… Và tất nhiên trong khóa học c bá đạo này, các bài tập và ví dụ là không thể thiếu trong mỗi bài học.

NỘI DUNG BÀI VIẾT

1. Video hướng dẫn hàm trong C
2. Hàm trong C là gì?
3. Ưu điểm khi dùng chương trình con
4. Cách hoạt động của hàm trong C
5. Chương trình máy tính bỏ túi đơn giản
6. Các loại hàm trong C
    6.1. 1. Hàm không có tham số, không có giá trị trả về
    6.2. 2. Hàm không có tham số, có trả về giá trị
    6.3. 3. Hàm có tham số, không trả về giá trị
    6.4. 4. Hàm có tham số, có trả về giá trị
7. Tài liệu tham khảo

Video hướng dẫn hàm trong C

Dưới đây là video hướng dẫn căn bản về hàm do người dùng định nghĩa ở trong ngôn ngữ C. Nói là **hàm người dùng định nghĩa** là để phân biệt với các hàm có sẵn trong các thư viện như bài học trước chúng ta vừa tìm hiểu. Các bạn nên tiếp tục đọc bài viết để có cái nhìn sâu sắc hơn về hàm trong C.

Hàm trong C là gì?

Đặt vấn đề: Giả sử bạn muốn xây dựng một công ty (một chương trình máy tính bỏ túi). Công ty của bạn được thành lập với 4 mục tiêu chính như sau: tính tổng, hiệu, tích, thương của 2 số nguyên nhập từ bàn phím.

Nếu công ty của bạn chỉ có một thành viên và thành viên đó phải tự làm và quản lý hết tất cả 4 công việc trên (code tất cả trong hàm `main()`) – Điều này là hoàn toàn khả thi. Nhưng bạn thử nghĩ xem, một mình bạn ôm cả 4 công việc đó thì liệu bạn có thể quản lý nó được tốt không? bạn có thể dành thời gian để tối ưu và phát triển mỗi công việc đó không? Bạn nghĩ sao nếu những lúc căng thẳng khiến bạn quên hoặc thực hiện nhầm công việc đáng ra mình cần làm?

Giải pháp: Thuê 4 ông nhân viên về và trả lương cho họ, mỗi ông (hàm con) chỉ làm một việc duy nhất. Khi đó, công việc của sếp (là hàm `main()` ý) là quản lý các ông nhân viên này, khi nào cần thì gọi ông ý làm cho mình và nếu có vấn đề gì ở 1 công việc nào đó thì cứ lôi cổ ông nhân viên đó ra mà xử lý.

Vậy thì hàm trong C là gì? Hàm chính là các ông nhân viên trong vấn đề phía trên. Trong lập trình, hàm là các khối code nhỏ chỉ thực hiện một chức năng nhất định của bài toán lớn.

Bạn có thể hình dung cái khung của công ty phía trên sau khi áp dụng giải pháp ta được như sau:

```
0
1  #include <stdio.h>
2
3  int add(int a, int b){
4     // hàm con này là nhân viên làm công việc phép cộng
5  }
6
7  int subtract (int a, int b){
8     // hàm con này là nhân viên làm công việc phép trừ
9  }
10
11 int mutiply (int a, int b){
12    // hàm con này là nhân viên làm công việc phép nhân
13 }
14
15 float devide (int a, int b){
16    // hàm con này là nhân viên làm công việc phép chia
17 }
18
```

```
19 int main(){
20     // hàm này là ông sếp
21 }
22
```

Ưu điểm khi dùng chương trình con

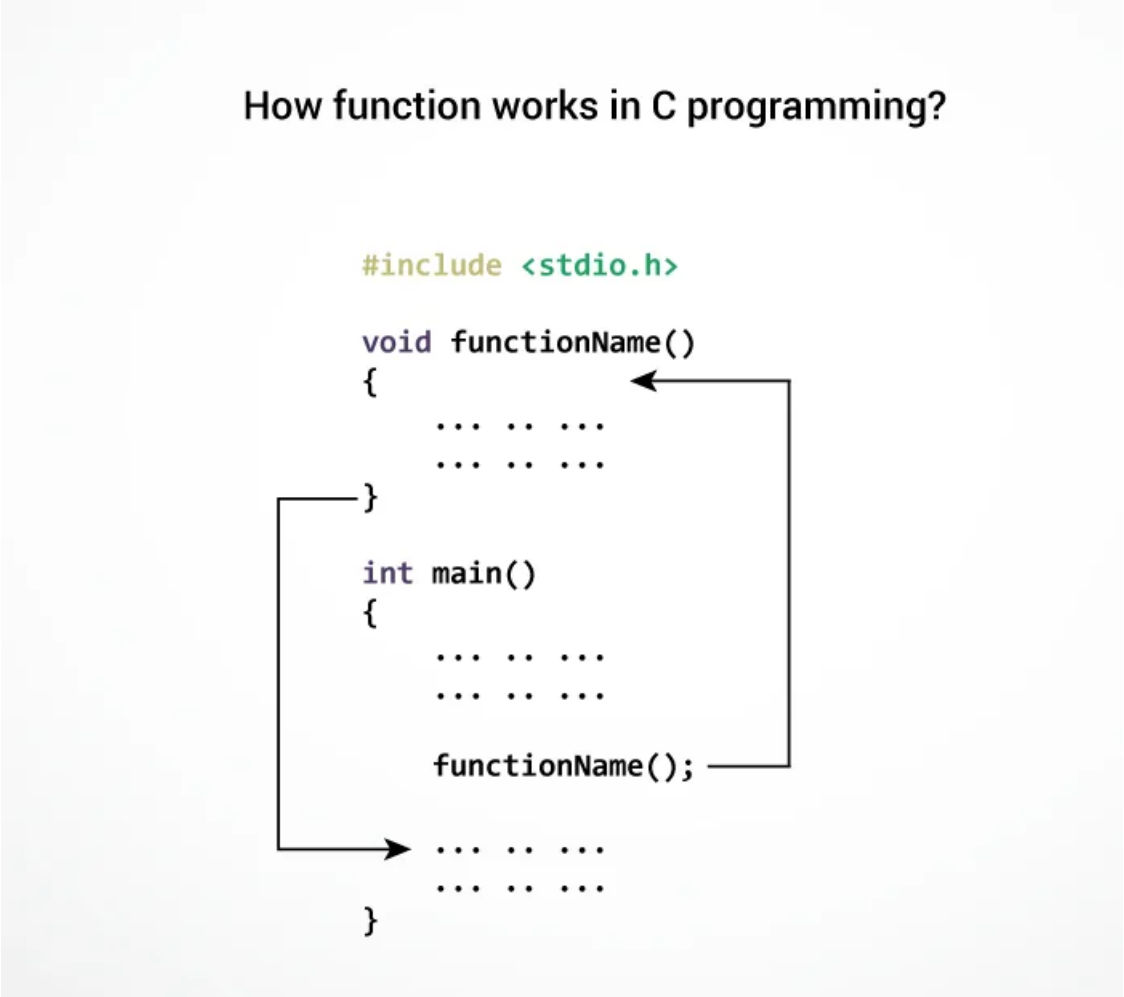
Sau đây là một số **ưu điểm nổi bật của sử dụng chương trình con (hàm)** mà mình có thể liệt kê, nhưng có 1 điều chắc chắn rằng: Hãy cố gắng thực hành viết code của bạn sử dụng hàm nếu có thể nhé.

- Sử dụng chương trình con khiến code của bạn trông sáng sủa hơn và gọn gàng, người đọc code sẽ dễ hiểu hơn bằng cách nhìn vào từng hàm con ta có thể dễ dàng xác định vai trò của nó trong chương trình.
- Dễ dàng quản lý, nâng cấp và tìm lỗi chương trình. Bởi vì bạn biết rõ hàm nào đang làm gì, nếu mà chẳng may gặp lỗi thì bạn cũng nhanh chóng xác định lỗi đó của hàm nào thay vì phải dò từng dòng trong hàm main
- Viết 1 lần và gọi được ở nhiều nơi: Khi bạn dùng hàm thì bạn chỉ phải viết một lần và gọi tới nó bất cứ khi nào bạn muốn. Bạn cũng có thể đóng gói các hàm đó để sử dụng cho các chương trình khác

Cách hoạt động của hàm trong C

Hình ảnh dưới đây cho bạn thấy cách hoạt động của hàm (chương trình con) ở trong ngôn ngữ C. Khi một lời gọi hàm được thực thi thì:

1. Chương trình của bạn sẽ nhảy tới nơi định nghĩa hàm đó và thực thi các lệnh từ trên xuống dưới ở trong hàm đó.
2. Khi hàm thực hiện xong, chương trình tiếp tục quay về thực hiện các lệnh phía sau lời gọi hàm.



Cách hoạt động của hàm ở trong ngôn ngữ C, ảnh: programiz.com

Chương trình máy tính bỏ túi đơn giản

Sau đây mình sẽ lấy một ví dụ sử dụng hàm trong C (chương trình con) để xây dựng ứng dụng máy tính bỏ túi đơn giản thực hiện 4 chức năng cơ bản là cộng, trừ, nhân, chia. Các bạn xem giải thích ở trong code cùng với xem video để hiểu hơn nhé.

```
0
1 #include <stdio.h>
2
```

```c
/*
    Cách khai báo hàm

    <kiểu dữ liệu> <tên hàm> (<các tham số>){
        // thân hàm
        return <value>; // value phải cùng kiểu <kiểu dữ liệu>
    }
 */
// Calculate sum of two number
// a: first number
// b: second number
int add(int a, int b){
    int sum = a + b;
    return sum; // lệnh return trả về giá trị có cùng kiểu dữ liệu với kiểu dữ liệu của hàm
}

// Hàm tính hiệu của a - b
int subtract(int a, int b){
    return a - b;
}

// Hàm tính tích a * b
int mutiply(int a, int b){
    return a * b;
}

// Hàm tính thương của a / b
// Lưu ý b cần != 0
// Bạn phải ép kiểu tử hoặc mẫu để thu được kết quả là số thực nhé
float devide(int a, int b){
    return (float) a / b;
}


int main(){
    int a = 3, b = 4;
    // Sử dụng tên hàm để gọi nó,
    // truyền các tham số đúng theo kiểu của hàm yêu cầu
    printf("\n%d + %d = %d", a, b, add(a, b));
    printf("\n%d - %d = %d", a, b, subtract(a, b));
    printf("\n%d * %d = %d", a, b, mutiply(a, b));
    if(b != 0){
        printf("\n%d / %d = %f", a, b, devide(a, b));
    }
```

3

```
}

```

Kết quả chạy chương trình:

```

PS G:\c_cources\day_27> g++ .\Calculator.cpp -o .\Calculator
PS G:\c_cources\day_27> .\Calculator.exe

3 + 4 = 7
3 - 4 = -1
3 * 4 = 12
3 / 4 = 0.750000

```

Các bài viết sau sẽ nói rõ hơn về các loại hàm và rất nhiều bài tập thực hành, bạn hãy tiếp tục theo dõi để trang bị cho mình kiến thức đầy đủ về hàm trong C nhé!

Các loại hàm trong C

Sau đây, mình sẽ lấy 1 ví dụ tham khảo từ tài liệu số [2] để thể hiện cho các bạn thấy có 4 loại hàm trong C. Chúng ta sẽ dùng 4 cách viết hàm khác nhau để giải quyết cùng 1 bài toán: "Kiểm tra 1 số người dùng nhập từ bàn phím có phải là số nguyên tố không". Sau cùng, chúng ta sẽ đi đến những kết luận!

1. Hàm không có tham số, không có giá trị trả về

```

#include <stdio.h>
void checkPrimeNumber();
int main()
{
    checkPrimeNumber();    // argument is not passed
    return 0;
}
// return type is void meaning doesn't return any value
void checkPrimeNumber()
{
    int n, i, flag = 0;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    for(i=2; i <= n/2; ++i)
```

```
    {
        if(n%i == 0)
        {
            flag = 1;
        }
    }
    if (flag == 1)
        printf("%d is not a prime number.", n);
    else
        printf("%d is a prime number.", n);
}

```

Như bạn thấy, hàm `checkPrimeNumber()` không có tham số đầu vào, bản thân nó tự thực hiện nhận giá trị từ bàn phím, kiểm tra và sau đó cũng in ra kết quả luôn. Vì là nó không trả về giá trị nên chúng ta dùng kiểu void, bạn sẽ học nó ở bài tiếp theo.

- Hàm này thực hiện 3 chức năng cùng 1 lúc => 1 hàm chỉ nên làm 1 chức năng

2. Hàm không có tham số, có trả về giá trị

```
#include <stdio.h>
int getInteger();
int main()
{
    int n, i, flag = 0;
   // no argument is passed
    n = getInteger();
    for(i=2; i<=n/2; ++i)
    {
        if(n%i==0){
            flag = 1;
            break;
        }
    }
    if (flag == 1)
        printf("%d is not a prime number.", n);
    else
        printf("%d is a prime number.", n);
    return 0;
}
// returns integer entered by the user
```

```
int getInteger()
{
    int n;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    return n;
}
```

Trong đoạn code trên, hàm `getInteger()` nhập 1 số từ bàn phím và trả ra cho chúng ta giá trị đó. Còn việc kiểm tra là số nguyên tố hay không thì chúng ta viết nó trong hàm `main()`. Tại vì chúng ta muốn viết hàm kiểm tra số nguyên tố thì hàm này cần tham số là "số cần kiểm tra".

- Như vậy, hàm `getInteger()` chỉ thực hiện 1 chức năng => ok. Nhưng hàm main vẫn phải đảm nhiệm công việc kiểm tra số nguyên tố (công việc nặng nhọc nhất)

3. Hàm có tham số, không trả về giá trị

```
#include <stdio.h>
void checkPrimeAndDisplay(int n);
int main()
{
    int n;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    // n is passed to the function
    checkPrimeAndDisplay(n);
    return 0;
}
// return type is void meaning doesn't return any value
void checkPrimeAndDisplay(int n)
{
    int i, flag = 0;
    for(i=2; i <= n/2; ++i)
    {
        if(n%i == 0){
            flag = 1;
            break;
        }
    }
    if(flag == 1)
        printf("%d is not a prime number.",n);
```

```
    else
        printf("%d is a prime number.", n);
}

```

Nhận thấy, hàm `checkPrimeAndDisplay()` nhận vào là một số cần kiểm tra, sau đó thực hiện kiểm tra và in ra kết quả.

- Hàm `checkPrimeAndDisplay()` đang làm 2 việc 1 lúc => cách viết hàm tốt thì 1 hàm chỉ làm 1 việc thôi.
- Lần này hàm main đảm nhiệm việc nhập (tốt) => Hàm main nên đảm nhận việc nhận input và xuất output.

4. Hàm có tham số, có trả về giá trị

```
0
#include <stdio.h>
int checkPrimeNumber(int n);
int main()
{
    int n, flag;
    printf("Enter a positive integer: ");
    scanf("%d",&n);
    // n is passed to the checkPrimeNumber() function
    // the returned value is assigned to the flag variable
    flag = checkPrimeNumber(n);
    if(flag == 1)
        printf("%d is not a prime number",n);
    else
        printf("%d is a prime number",n);
    return 0;
}
// int is returned from the function
int checkPrimeNumber(int n)
{
    int i;
    for(i=2; i <= n/2; ++i)
    {
        if(n%i == 0)
            return 1;
    }
    return 0;
```

```
27 }
28
```

Ở trường hợp lần này, hàm `checkPrimeNumber()` chỉ nhận nhiệm vụ nhận vào 1 số và kiểm tra xem số đó có phải số nguyên tố hay không.

- Hàm main đảm nhận nhiệm vụ lấy đầu vào, xuất kết quả => Tuyệt vời
- Hàm con `checkPrimeNumber()` chỉ làm 1 việc duy nhất => Tuyệt vời

Mình chót lỡ nhận xét sau mỗi ví dụ rồi thì thôi các bạn tự điền câu chốt hạ giúp mình nhé. Việc sử dụng hàm cần khéo léo để code của chúng ta được "sạch sẽ". Các bạn nhớ đọc thêm cả tài liệu tham khảo nữa nhé!

Tài liệu tham khảo

1. https://www.programiz.com/c-programming/c-functions
2. https://www.programiz.com/c-programming/types-user-defined-functions

Các bài viết trong khóa học

Bài trước: Bài 26. Thư viện math.h trong C

Bài sau: Bài 28. Tìm max min của 3 số a b c nhập từ bàn phím

3

Nguyễn Văn Hiếu

Sáng lập cộng đồng Lập Trình Không Khó với mong muốn giúp đỡ các bạn trẻ trên con đường trở thành những lập trình viên tương lai. Tất cả những gì tôi viết ra đây chỉ đơn giản là sở thích ghi lại các kiến thức mà tôi tích lũy được.

Blog chia sẻ kiến thức lập trình của Hiếu, xây dựng cộng đồng những người học lập trình. Cho đi kiến thức mình có là cách học tập hiệu quả nhất

👉 Báo lỗi / Liên hệ / Hợp tác / Quảng cáo

BÀI VIẾT HAY

Bài 1. Giới thiệu khóa học "Học C Bá Đạo"
21/07/2019

1000 bài tập lập trình C/C++ có lời giải của thầy Khang
25/12/2019

Kiểm tra số nguyên tố sử dụng C/C++ và Java
15/07/2018

CHUYÊN MỤC HAY

Học C/C++	194
Học Python	48
Học Java	45
Học Javascript	37
Khóa học	33
Học Web	26
Học C#	24
Chia sẻ	23

- BẠN BÈ & ĐỐI TÁC -

Luyện Code - Tự Học Đồ Họa - Cách Học Lập Trình - VNTALKING

3