I. Tổng quan về hàm/ chương trình con

1. Định nghĩa hàm

Quan sát ví dụ sau đây:

```
1   #include <stdio.h>
2   int addition(int a, int b);
3
4   int main(){
5       printf("The result is : %d",addition(5,3));
6       return 0;
7   }
8
9   int addition(int a, int b){
10      int r;
11      r = a + b;
12      return r;
13  }
```

Kết quả xuất ra màn hình của ví dụ này là câu thông báo: **"The result is : 8"**

Trong ví dụ trên, ta chú ý đến 3 đoạn mã: đoạn lệnh số 2 (1) được gọi là ***prototype***, đoạn lệnh từ 4 đến 7 (2) và đoạn lệnh từ 9 đến 12 (3) được gọi là hàm ***(function) hay chương trình con (subprogram).*** Mỗi hàm có cú pháp như sau:

type name (parameter1, parameter2,..)
{
statement;
}

- ***name:*** tên của hàm. Tên cũng là một định danh ***(identifier).*** Do đó tên cần được đặt tuân theo qui tắt đặt tên của ngôn ngữ lập trình. Trong ví dụ trên, chức năng của hàm là cộng hai số nguyên nên tên hàm được đặt là ***addition.***

- ***type:*** kiểu dữ liệu trả về của hàm. Trong ví dụ trên, kiểu dữ liệu trả về của hàm addition là kiểu ***int.*** Một vài trường hợp, nếu một hàm không cần giá trị trả về thì các bạn dùng kiểu ***void***

- ***parameters:*** danh sách tham số của hàm. Một hàm có ít nhất 0 tham số, và nhiều nhất là n tham số tùy theo từng bài toán. Các tham số cách nhau bởi dấu (‘,’), được bao trogn dấu ‘(‘ và ‘)’. Mỗi tham số tuân theo quy tắt khai báo biến: ***type name.*** Ví dụ trên thì danh sách tham số được khai báo là ***(int a, int b)***

- ***statements:*** là câu lệnh được viết ra nhằm mục đích hiện thực chức năng của hàm, được bao bọc bởi dấu ‘{‘ và ‘}’. Trong ví dụ trên phần statement của hàm addition là đoạn mã lệnh từ 10

đến 12. Phần statements có thể thuộc một trong các loại sau: ***cấu trúc điều khiển – control structures (cấu trúc tuần tự - sequence structure, cấu trúc rẽ nhánh – selection structure: if, if .. else và cấu trúc lặp – iteration structure)*** hoặc lời gọi đến một function, hoặc thậm chí là lời gọi đến chính function đó mà ta gọi là ***đệ qui (recursion)***

- ***return:*** là từ khóa trong C/C++, từ khóa này làm cho một hàm kết thúc ngay lập tức và trả về giá trị kiểu dữ liệu phù hợp với ***type*** đã định nghĩa. Trong ví dụ trên, từ return tại lệnh số 12 trả về giá trị có kiểu ***int*** là kết quả của phép cộng hai số a và b, đồng thời kết thúc hàm ***addition***

Lưu ý: hàm main cũng là một hàm. Hàm main là một hàm đặc biệt trong C, là điểm bắt đầu chương trình. Một chương trình chỉ có duy nhất một hàm main.

2. Lời gọi hàm và truyền đối số

Lời gọi hàm (function call) là việc một chương trình hay một hàm nào đó gửi yêu cầu đến một hàm khác để hỗ trợ nó hoàn thành chức năng của mình. Hàm được gọi bắt buộc phải được định nghĩa trước đó tại một nơi nào đó của chương trình nhưng không có điều ngược lại. Nghĩa là, một hàm được định nghĩa rồi thì có thể không được gọi trong toàn bộ chương trình. Khi đó việc định nghĩa hàm trở nên vô nghĩa.

Một hàm có thể được gọi nhiều lần để tái sử dụng. Trong ví dụ trên, hàm main gọi hàm addtion để thực hiện cộng hai số. Cú pháp một lời gọi hàm tuân theo quy tắc như sau:

name(argument1, argument2,..);

Trong đó:

- ***name:*** là tên hàm muốn gọi. Tên này phải đúng như tên đã được định nghĩa trước đó.

- ***argument:*** là danh sách các đối số truyền cho hàm để hảm thực thi. Xem ví dụ sau để hiểu về cách gọi hàm và truyền tham số:

```c
int main()
{
    printf("The result is : %d \n",addition(5,3));

    int x = 0, y = 0;
    printf("Moi ban nhap so thu nhat ");
    scanf("%d",&x);
    printf("Moi ban nhap so thu hai ");
    scanf("%d",&y);

    // Goi ham addition va gan gia tri x cho a, y cho b
    printf("The result is: %d \n",addition(x,y));

    // Goi ham addition va gan gia tri y cho a, x cho b
    printf("The result is: %d \n",addition(y,x));

    return 0;
}
```

Ví dụ trên, hàm addition được hàm main gọi 3 lần. Mỗi lần hàm addition thực thi, các đối số a và b mang giá trị khác nhau. Lưu ý: đến thứ tự của các đối số vì việc này sẽ ảnh hưởng đến chương trình. Với chương trình trên giả sử nhập số thứ nhất là 2 và số thứ hai là 6 thì kết quả đều ra 8.

Bài tập nhỏ áp dụng

a. Viết hàm thực hiện **- , *, /,** tìm max hai số. Đối với phép chia có kiểm tra nếu mẫu số bằng 0 thì in thông báo lỗi và kết thúc hàm.

b. Viết hàm tính a^x, với a và x đều là số nguyên dương

c. Viết hàm main gọi để kiểm tra tính đúng đắn của các hàm vừa viết.

3. Đệ qui – Recursion

Như đã nói ở trên, các hàm có thể thực hiện gọi một hoặc nhiều hàm khác nhau để hỗ trợ hàm đó thực hiện chức năng của mình. Khi một hàm gọi lại chính nó, ta có được khái niệm đệ qui. Hiểu theo 1 cách mộc mạc, đệ qui giống như bên trong một chiếc hộp có một chiếc hộp khác tương tự nhưng kích thước nhỏ hơn và cứ tiếp tục đến chiếc hộp bé nhất. Ta mở hộp nhỏ nhất ra và nhận được một món quà ý nghĩa. Xét ví dụ về bài tính n! đã được học ở các buổi trước

```c
#include <stdio.h>
int giaithua_1(int); //prototype
int giaithua_2(int); //prototype
int giaithua_3(int); //prototype

int main()
{
    printf("Cach 1: Giai thua cua 4 la: %d \n",giaithua_1(4));
    printf("Cach 2: Giai thua cua 4 la: %d \n",giaithua_2(4));
    printf("Cach 3: Giai thua cua 4 la: %d \n,giaithua_3(4));
    return 0;
}

int giaithua_1(int n)
{
    int i, gt = 1;
    for(i = 2; i<=n;i++){
        gt = gt * i;
    }
    return gt;
}
int giaithua_2(int n)
{
    if( n == 1 || n == 0){
        return 1;
    }
    else
    {
        return n * giaithua_2(n-1);
    }
}
int giaithua_3(int n)
{
    return n > 1 ? n * giaithua_3(n-1): 1;
}
```

4. **Tổ chức chương trình:** mỗi một hàm chỉ nên thực hiện một chức năng duy nhất, không nên gộp nhiều chức năng vào trong một hàm.

II. Bài tập

1. Viết hàm:

a) Tính chu vi và diện tích của hình tròn

b) Tính chu vi và diện tích của hình chữ nhật

c) Tìm số lớn nhất trong 3 số cho trước.

d) Tìm tổng của S2 = 1 + 3 + 5 + … + (2n+1)

e) S5 = 13 + 33 + 53 + … + (2n+1)3

f) Viết hàm main để kiểm chứng các hàm trên

2. Sử dụng hàm để viết chương trình có chức năng sau đây:

a) Cho phép người dùng nhập 3 số thực đại diện cho độ dài của 3 cạnh một tam giác. Nếu độ dài không thỏa mãn điều kiện hình thành tam giác thì yêu cầu người dùng nhập lại cho đến khi thỏa mãn.

b) Kiểm tra thuộc tính tam giác: vuông, cân, đều hay bình thường

c) Tính và in ra màn hình chu vi và diện tích của tam giác, biết $S = \sqrt{p(p-a)(p-b)(p-c)}$ với p = (a+b+c)/2.

d) Viết hàm main để kiểm chứng

3. Viết hàm cho phép người dùng nhập vào một số nguyên dương n (n > 0). Chương trình kiểm tra nếu số n không nguyên dương thì yêu cầu người dùng nhập lại. Khi số thỏa điều kiện thì viết hàm:

a) Kiểm tra xem n có phải là một số nguyên tố hay không ?

b) Kiểm tra xem n có phải là một số toàn chẵn hay không ?

c) Kiểm tra xem n có phải là số toàn lẽ hay không ?

d) Kiểm tra xem n có phải là chữ số tăng dần từ trái qua phải hay không ?

e) Kiểm tra xem n có phải là chữ số giảm dần từ trái qua phải hay không ?

f) Kiểm tra xem n có phải là số đối xứng hay không ?

g) Viết hàm main để kiểm chứng.

4. Viết hàm cho phép người dùng nhập vào một hai số nguyên dương a và b (a,b > 0). Chương trình kiểm tra nếu hai số a và b không nguyên dương thì yêu cầu người dùng nhập lại. Khi số thỏa điều kiện thì viết hàm:

a) Tìm và in tất cả ước chung của hai số

b) Viết hàm tìm ra số lớn nhất trong 2 số.

c) Viết hàm thực hiện hoán đổi giá trị của hai số vừa nhập từ bàn phím.

d) Viết hàm main để kiểm chứng.

5. Viết hàm sử dụng đệ qui để tính:

a) Viết hàm đệ qui tìm ước chung lớn nhất của 2 số nguyên dương

$$UCLN(m, n) = \begin{cases} m & ,nếu\ n = 0 \\ UCLN(n, m\%n) & ,nếu\ n > 0 \end{cases}$$

b) Viết hàm đệ qui tính X mũ n (X^n) với X là số thực, n là số nguyên.

$$x^n = \begin{cases} 1 & ,nếu\ n = 0 \\ x^{n-1}x & ,nếu\ n > 0 \end{cases}$$

c) Viết hàm đệ qui tính tổ hợp chập k của n, được xác định như sau:

$$C(n,k) = \begin{cases} C(n,k) = 1, \text{nếu k = 0 hoặc k = n} \\ C(n,k) = C(n-1,k) + C(n-1,k-1), \text{ngược lại} \end{cases}$$

d) Viết hàm đệ qui tính tổng S = 1 + 2 + 3 + … + n

e) Viết hàm đệ qui tính tổng S = 1 + 1/2 + 1/3 + … + 1/n

f) Viết hàm đệ qui tính tổng chẵn lẽ n số tự nhiên. Nếu n chẵn thì TONGCHANLE(n) = 2 + 4 + .. + n, còn nếu n lẽ thì TONGCHANLE(n) = 1 + 3 + … + n.