TRƯỜNG ĐẠI HỌC KINH TẾ QUỐC DÂN
VIỆN CÔNG NGHỆ THÔNG TIN VÀ KINH TẾ SỐ

CƠ SỞ LẬP TRÌNH

CHƯƠNG 6: HÀM TRONG C

huongct@neu.edu.vn

MỤC TIÊU

- Hiểu về khái niệm hàm, cấu trúc của một hàm, cách sử dụng hàm
- Liệt kê và phân biệt được các loại biến, các loại tham số của hàm
- Biết cách gọi hàm
- Phân biệt về các cách truyền tham số
- Vận dụng hàm trong các bài toán đệ quy

CHƯƠNG 6: HÀM TRONG C

- 6.1. Khái niệm hàm trong C
- 6.2. Khai báo và sử dụng hàm
- 6.3. Các cách truyền tham số trong hàm
- 6.4. Phân loại biến sử dụng trong chương trình
- 6.5. Nguyên mẫu hàm trong C
- 6.6. Hàm đệ quy

CHƯƠNG 6: HÀM TRONG C

- 6.1. Khái niệm hàm trong C
- 6.2. Khai báo và sử dụng hàm
- 6.3. Các cách truyền tham số trong hàm
- 6.4. Phân loại biến sử dụng trong chương trình
- 6.5. Nguyên mẫu hàm trong C
- 6.6. Hàm đệ quy

6.1. KHÁI NIỆM HÀM TRONG C

- Hàm là một khối lệnh **chỉ chạy** khi nó **được gọi**.
- Có thể truyền dữ liệu (tham số-parameters) vào một hàm.
- Các hàm được sử dụng để thực hiện các hành động nhất định và rất quan trọng để sử dụng lại mã lệnh:
  - **Viết** mã lệnh **một lần** và **sử dụng** nó **nhiều lần**.

6.1. KHÁI NIỆM HÀM TRONG C

- Các hàm được xác định trước
  - Ví dụ:

```
#include <stdio.h>

void main() {
  printf("Hello World!");
}
```

  - `main()` là một hàm, được sử dụng để thực thi mã lệnh
  - `printf()` là một hàm; được sử dụng để xuất/in dữ liệu ra màn hình

6.1. KHÁI NIỆM HÀM TRONG C

- Trong C, một số hàm (chương trình con) thông dụng đã được thực hiện sẵn như **sin, cos, pow, sqrt**...
- Những chương trình con này nằm trong *thư viện các chương trình con mẫu*, do trình biên dịch C quản lý → còn được gọi là các chương trình con chuẩn.

6.1. KHÁI NIỆM HÀM TRONG C

- Một số hàm toán học: (**#include <math.h>**)

Hàm	Mô tả
sqrt(x)	Trả lại căn bậc hai của x
abs(x)	Trả lại trị tuyệt đối của x
ceil(x)	Trả lại số nguyên gần nhất của x (>=x)
floor(x)	Trả lại số nguyên gần nhất của x (<=x)
pow(x, y)	Trả lại x^y

Hàm	Mô tả
acos(x)	Trả lại arccosine của x
asin(x)	Trả lại arcsine của x
atan(x)	Trả lại arctangent của x
cbrt(x)	Trả lại cube root của x
cos(x)	Trả lại cosine của x
exp(x)	Trả lại value của E^x
sin(x)	Trả lại sine của x (x theo radians)
tan(x)	Trả lại tangent của một góc

6.1. KHÁI NIỆM HÀM TRONG C

- Việc chia nhỏ một chương trình thành các chương trình con đảm nhận những công việc nhỏ khác nhau → tưởng chính cho phương pháp lập trình có cấu trúc (*structured programming*).
- Lưu ý:
  - Khi một chương trình con chỉ sử dụng đúng một lần nhưng nó vẫn làm cho chương trình trở nên sáng sủa, dễ đọc, dễ hiểu hơn.

CHƯƠNG 6: HÀM TRONG C

- 6.1. Khái niệm hàm trong C
- 6.2. Khai báo và sử dụng hàm
- 6.3. Các cách truyền tham số trong hàm
- 6.4. Phân loại biến sử dụng trong chương trình
- 6.5. Nguyên mẫu hàm trong C
- 6.6. Hàm đệ quy

6.2. KHAI BÁO VÀ SỬ DỤNG HÀM

6.2.1. Khai báo hàm

6.2.2. Gọi hàm

6.2.1. KHAI BÁO HÀM

- Cú pháp:

```
returnType functionName(parameter1, parameter2, parameter3,…)
{
  // code to be executed
  return value;
}
```

  - Trong thân hàm có lệnh **return** để trả về giá trị cho hàm

6.2.1. KHAI BÁO HÀM

- Ví dụ: hàm tính giai thừa của một số nguyên dương.
  - Quy ước: n!=−1 (nếu n<0), n!=0 (nếu n=0), n!=1*2*3*…*n (với n ≠0)

```
int giai_thua(int a)
{
    int i, kq = 1;
    if(a < 0)
        kq = -1;
    else if(a == 0)
        kq = 0;
    else
        for(i = 1; i <= a; i++)
            kq = kq * i;
    return kq;
}
```

→ Khai báo hàm

Thân hàm

→ giá trị trả về cho hàm

6.2.1. KHAI BÁO HÀM

```
returnType functionName(parameter1, parameter2, parameter3,…)
{
  // code to be executed
   return value;
}
```

❖ ***returnType*: Kiểu dữ liệu trả về của hàm**

- Trong C, kiểu dữ liệu trả về của hàm có thể là kiểu dữ liệu bất kì (kiểu dữ liệu có sẵn hoặc kiểu dữ liệu do người dùng tự định nghĩa) nhưng **không được** là kiểu dữ liệu **mảng**.
- Nếu kiểu dữ liệu trả về là kiểu **void** thì hàm không trả về giá trị nào.
- Trường hợp **không khai báo** kiểu dữ liệu trả về thì chương trình dịch của C sẽ mặc định kiểu dữ liệu trả về của hàm là kiểu **int**.

6.2.1. KHAI BÁO HÀM

```
returnType functionName(parameter1, parameter2, parameter3,…)
{
  // code to be executed
  return value;
}
```

❖ ***functionName***: **Tên hàm**

- Tên hàm đặt theo yêu cầu của định danh.
- Tên hàm nên mang nghĩa **gợi ý chức năng** công việc mà hàm thực hiện.
- Trong C, các hàm **không** được đặt tên **trùng** nhau.

6.2.1. KHAI BÁO HÀM

```
returnType functionName(parameter1, parameter2, parameter3,…)
{
  // code to be executed
   return value;
}
```

❖***parameter*: Tham số của hàm**

- Chứa dữ liệu vào cung cấp cho hàm, hoặc chứa dữ liệu ra mà hàm tính toán được.
- Các tham số sử dụng trong lời khai báo hàm được gọi là tham số hình thức (tham số giả định).
- Các tham số được cung cấp cho hàm trong quá trình thực hiện được gọi là tham số thực (đối số).
- Kiểu dữ liệu của **tham số thực** cung cấp cho hàm trong chương trình phải **giống *kiểu dữ liệu*** của **tham số hình thức.**

6.2.1. KHAI BÁO HÀM

- Ví dụ

```
#include <stdio.h>
int timMax(int x, int y)
{    int m;
     if (x > y) m = x;
     else m = y;
     return m;
}
void main()
{    int a, b, c, max;
     printf("\nNhap vao 3 so:");
     scanf("%d%d%d", &a, &b, &c);
     max = timMax(a, b);
     max = timMax(c, max);
     printf("\nSo lon nhat trong 3 so da nhap la: %d", max);
}
```

Tham số hình thức

Tham số thực sự (đối số)

6.2.1. KHAI BÁO HÀM

```
returnType functionName(parameter1, parameter2, parameter3,…)
{
  // code to be executed
  return value;
}
```

❖ ***Parameter:*** **Tham số của hàm**

- Một hàm có thể có một, nhiều hoặc không có tham số nào cả.
- Nếu có nhiều tham số thì chúng phải được phân cách với nhau bằng dấu phẩy.
- Nếu hàm không có tham số nào cả thì vẫn phải có cặp dấu ngoặc đơn sau tên hàm, ví dụ **main()**.

6.2.1. KHAI BÁO HÀM

- **Lệnh return**
  - Sau khi hàm thực hiện xong, nó sẽ trở về chương trình đã gọi nó.
  - Có 2 cách để từ hàm trở về chương trình đã gọi hàm:
    - Sau khi thực hiện tất cả các câu lệnh có trong thân hàm.
    - Khi gặp lệnh return.

6.2.1. KHAI BÁO HÀM

- Lệnh **return**:
  - Lệnh **return** ngay lập tức chuyển điều khiển từ hàm trở về chương trình gọi.
  - Giá trị đặt sau lệnh **return** được trả về cho chương trình gọi.

```
int timMax(int x, int y)
{    int m;
     if (x > y) m = x;
     else m = y;
     return m;
}
```

6.2.1. KHAI BÁO HÀM

Lệnh return

- Cú pháp:

```
return [biểu_thức];
```

- Khi gặp lệnh này, chương trình sẽ:
  - Tính toán giá trị của **biểu_thức**
  - Lấy kết quả tính toán được **làm giá trị trả về cho lời gọi hàm**
  - Kết thúc việc thực hiện hàm, trở về chương trình đã gọi nó.
- Trong lệnh **return** cũng có thể không có phần **biểu_thức**,

  → kết thúc thực hiện hàm mà **không trả về giá trị** nào cả.

6.2.1. KHAI BÁO HÀM

- Ví dụ:

```
#include <stdio.h>
#include <conio.h>
int max(int x, int y, int z)
{
    int max;
    max = x>y?x:y;
    max = max>z?max:z;
    return max;
}
void main()
{
    int a,b,c;
    printf("\n Nhap gia tri cho 3 so nguyen a, b, c: ");
    scanf("%d %d %d",&a,&b,&c);
    printf("\n Gia tri cac so vua nhap: ");
    printf("a = %-5d b = %-5d c = %-5d",a,b,c);
    printf("\n Gia tri lon nhat trong 3 so la %d",max(a,b,c));
    getch();
}
```

6.2.2. GỌI HÀM

- Các hàm đã khai báo không được thực thi ngay lập tức.
- Chúng được "lưu để sử dụng sau", và sẽ được thực thi khi chúng **được gọi**.
- Cách gọi hàm: viết tên hàm theo sau bởi hai dấu ngoặc đơn () và dấu chấm phẩy;

6.2.2. GỌI HÀM

- Ví dụ: Trong hàm `main`, gọi hàm `myFunction()`:
  - **myFunction**() được sử dụng để in một văn bản (hành động), khi nó được gọi

```
#include <stdio.h>

// Create a function
void myFunction() {
  printf("I just got executed!");
}

int main() {
  myFunction(); // call the function
  return 0;
}
```

```
I just got executed!
```

6.2.2. GỌI HÀM

- Ví dụ: Một hàm có thể được gọi nhiều lần

```c
#include <stdio.h>

// Create a function
void myFunction() {
  printf("I just got executed!\n");
}

int main() {
  myFunction(); // call the function
  myFunction(); // call the function
  myFunction(); // call the function
  return 0;
}
```

```
I just got executed!
I just got executed!
I just got executed!
```

6.2.2. GỌI HÀM

- Tham số (Parameters) và đối số (Arguments)
  - Thông tin có thể được truyền cho các hàm dưới dạng một tham số.
  - Các tham số hoạt động như các biến bên trong hàm.
  - Các tham số được chỉ định sau tên hàm, bên trong dấu ngoặc đơn.
  - Số lượng tham số là túy ý, các tham số phân tách nhau bằng dấu phẩy
- Khi một tham số được truyền cho hàm, nó được gọi là một đối số.

6.2.2. GỌI HÀM

- Ví dụ: Hàm **myFunction** nhận một chuỗi ký tự **name** làm tham số.
  - **name** là một **tham số**
  - **Liam**, **Jenny** và **Anja** là các **đối số**.

```
Hello Liam
Hello Jenny
Hello Anja
```

```c
#include <stdio.h>

void myFunction(char name[]) {
  printf("Hello %s\n", name);
}

int main() {
  myFunction("Liam");
  myFunction("Jenny");
  myFunction("Anja");
  return 0;
}
```

6.2.2. GỌI HÀM

- Bên trong hàm, có thể có nhiều tham số.
  - Ví dụ:

```c
#include <stdio.h>

void myFunction(char name[], int age) {
  printf("Hello %s. You are %d years old\n", name, age);
}

int main() {
  myFunction("Liam", 3);
  myFunction("Jenny", 14);
  myFunction("Anja", 30);
  return 0;
}
```

```
Hello Liam. You are 3 years old
Hello Jenny. You are 14 years old
Hello Anja. You are 30 years old
```

6.2.2. GỌI HÀM

- Ví dụ:

```
Result is: 8
```

```c
#include <stdio.h>

int myFunction(int x) {
  return 5 + x;
}

int main() {
  printf("Result is: %d", myFunction(3));
  return 0;
}
```

6.2.2. GỌI HÀM

- Ví dụ: hàm trả lại tổng của 2 tham số

```
Result is: 8
```

```c
#include <stdio.h>

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

int main() {
  printf("Result is: %d", myFunction(5, 3));
  return 0;
}
```

6.2.2. GỌI HÀM

- Ví dụ:

```
Result is: 8
```

```c
#include <stdio.h>

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

int main() {
  int result = myFunction(5, 3);
  printf("Result is = %d", result);
  return 0;
}
```

6.2.2. GỌI HÀM

- Dấu chấm phẩy được đặt cuối câu lệnh khi gọi hàm, nhưng không dùng cho định nghĩa hàm.
- Cặp dấu ngoặc () là bắt buộc theo sau tên hàm, cho dù hàm có đối số hay không.
- Hàm có nhiều nhất một giá trị được trả về.
- Chương trình có thể có nhiều hơn một hàm.
- Hàm gọi đến một hàm khác được gọi là **hàm gọi.**
- Hàm đang được gọi đến được gọi là **hàm được gọi.**

CHƯƠNG 6: HÀM TRONG C

- 6.1. Khái niệm hàm trong C
- 6.2. Khai báo và sử dụng hàm
- 6.3. Các cách truyền tham số trong hàm
- 6.4. Phân loại biến sử dụng trong chương trình
- 6.5. Nguyên mẫu hàm trong C
- 6.6. Hàm đệ quy

6.3. CÁC CÁCH TRUYỀN THAM SỐ TRONG HÀM

- Các kiểu truyền tham số cho hàm
  - Truyền bằng giá trị
  - Truyền tham chiếu

6.3.1. TRUYỀN THAM SỐ CHO HÀM BẰNG GIÁ TRỊ

- Trong C, mặc định tất cả các đối số được truyền bằng giá trị
- Các đối số được gọi là truyền bằng giá trị khi **giá trị của biến** được truyền đến hàm được gọi thông qua các biến tạm. Mọi sự thao tác chỉ được thực hiện trên các biến tạm

→ Bất kỳ sự thay đổi trên giá trị này (trên biến tạm) **không ảnh hưởng** đến *giá trị gốc của biến được truyền.*

6.3.1. TRUYỀN THAM SỐ CHO HÀM BẰNG GIÁ TRỊ

- Ví dụ

```
#include <stdio.h>
 void swap(int a, int b)
{
    int temp;
    temp = a;
    a = b;
    b = temp;
}
void main()
{
    int x, y;
    x= 10; y = 50;
    printf("\nTruoc khi doi cho x= %d, y = %d\n", x, y);
    swap(x, y);
    printf("\nSau khi doi cho x = %d, y = %d\n", x, y);
}
```

```
Truoc khi doi cho x= 10, y = 50

Sau khi doi cho x = 10, y = 50
```

6.3.2. TRUYỀN THAM SỐ CHO HÀM BẰNG THAM CHIẾU

- Với truyền tham chiếu, hàm cho phép truy xuất đến địa chỉ thực trong bộ nhớ của đối số.
- → có thể thay đổi giá trị của các đối số của hàm gọi.

6.3.2. TRUYỀN THAM SỐ CHO HÀM BẰNG THAM CHIẾU

- Ví dụ:

6.3.2. TRUYỀN THAM SỐ CHO HÀM BẰNG THAM CHIẾU

- Ví dụ:

```
#include <stdio.h>
 void swap(int *a, int *b)
{
    int temp;
    temp = *a;
    *a = *b;
    *b = temp;
}
void main()
{
    int x, y;
    x= 10; y = 50;
    printf("\nTruoc khi doi cho x= %d, y = %d\n", x, y);
    swap(&x, &y);
    printf("\nSau khi doi cho x = %d, y = %d\n", x, y);
}
```

```
Truoc khi doi cho x= 10, y = 50

Sau khi doi cho x = 50, y = 10
```

6.3.2. TRUYỀN THAM SỐ CHO HÀM BẰNG THAM CHIẾU

- Ví dụ:

```
nhap vao so n = 7

7! = 5040
```

```
#include<stdio.h>
void tinhGt(int a , int *b){
    int i,gt;
    gt=1;
    for (i=1;i<=a;i++)
        gt*=i;
    if(a<0)
        gt=-1;
    if(a==0)
        gt=0;
    *b=gt;
}
void main(){
    int n,kq;
    printf("nhap vao so n = "); scanf("%d",&n);
    tinhGt(n,&kq);
    printf("\n%d! = %d",n,kq);
}
```

CHƯƠNG 6: HÀM TRONG C

- 6.1. Khái niệm hàm trong C
- 6.2. Khai báo và sử dụng hàm
- 6.3. Các cách truyền tham số trong hàm
- 6.4. Phân loại biến sử dụng trong chương trình
- 6.5. Nguyên mẫu hàm trong C
- 6.6. Hàm đệ quy

6.4. PHÂN LOẠI BIẾN SỬ DỤNG TRONG CHƯƠNG TRÌNH

- Phạm vi của các biến
  - Nguyên tắc sử dụng biến là ***biến khai báo trong phạm vi nào thì được sử dụng trong phạm vi đó***.
  - Một biến có thể được khai báo trong chương trình chính hoặc trong các chương trình con hoặc thậm chí trong một lệnh khối.
    - Nếu biến được khai báo trong một lệnh khối thì nó chỉ có thể được gọi trong lệnh khối đó thôi, không thể gọi từ bên ngoài lệnh khối được
    - Tương tự cho biến khai báo trong CTC và ctc.

6.4. PHÂN LOẠI BIẾN SỬ DỤNG TRONG CHƯƠNG TRÌNH

- Việc trùng tên của các biến:
  - Trong cùng một phạm vi ta không được phép khai báo 2 biến có cùng tên nhưng ta có thể khai báo 2 biến trùng tên thuộc 2 phạm vi khác nhau.
  - Nếu có 2 biến trùng tên khai báo ở 2 phạm vi khác nhau thì:
    - Hai phạm vi này **tách rời nhau**: khi đó các biến sẽ có tác dụng ở phạm vi riêng của nó, không ảnh hưởng đến nhau.
    - Phạm vi này **nằm trong** phạm vi kia: khi đó nếu chương trình đang ở phạm vi ngoài (tức là đang thực hiện câu lệnh nằm ở phạm vi ngoài) thì biến khai báo ở phạm vi ngoài có tác dụng, còn nếu chương trình đang ở phạm vi trong (đang thực hiện câu lệnh nằm ở phạm vi trong) thì biến khai báo ở phạm vi trong sẽ có tác dụng và nó che lấp biến trùng tên ở bên ngoài.

6.4. PHÂN LOẠI BIẾN SỬ DỤNG TRONG CHƯƠNG TRÌNH

- Ví dụ:

```
a = 1
a = 2
a = 1
a = 3
```

```c
#include <stdio.h>
void main()
{
    {
        int a = 1;
        printf("\n a = %d",a);
        {
            int a = 2;
            printf("\n a = %d",a);
        }
        printf("\n a = %d",a);
    }
    {
        int a = 3;
        printf("\n a = %d",a);
    }
}
```

6.4. PHÂN LOẠI BIẾN SỬ DỤNG TRONG CHƯƠNG TRÌNH

- Biến địa phương hay cục bộ (Local Variable):
  - Là các biến được khai báo trong lệnh khối hoặc trong thân chương trình con.
- Biến toàn cục (Global Variable):
  - Là biến được khai báo sau phần khai báo tệp tiêu đề và khai báo hàm nguyên mẫu

6.4. PHÂN LOẠI BIẾN SỬ DỤNG TRONG CHƯƠNG TRÌNH

- Ví dụ: Biến cục bộ

```
a = 1
a = 2
a = 1
a = 3
```

```c
#include <stdio.h>
void main()
{
    {
        int a = 1;
        printf("\n a = %d",a);
        {
            int a = 2;
            printf("\n a = %d",a);
        }
        printf("\n a = %d",a);
    }
    {
        int a = 3;
        printf("\n a = %d",a);
    }
}
```

6.4. PHÂN LOẠI BIẾN SỬ DỤNG TRONG CHƯƠNG TRÌNH

- Ví dụ: Biến toàn cục

```
#include <stdio.h>
#include <conio.h>
int a, b, c;
int tich()
{   printf("\n Gia tri cac bien tong the a, b, c: ");
    printf(" a = %-5d b = %-5d c = %-5d", a, b, c);
    return a*b*c;
}
void main()
{
    printf("\n Nhap gia tri cho 3 so nguyen a, b, c: ");
    scanf("%d %d %d",&a,&b,&c);
    printf("\n Tich cua 3 so la %d",tich(a,b,c));
}
```

```
Nhap gia tri cho 3 so nguyen a, b, c: 5 8 2

Gia tri cac bien tong the a, b, c:  a = 5     b = 8     c = 2
Tich cua 3 so la 80
```

CHƯƠNG 6: HÀM TRONG C

- 6.1. Khái niệm hàm trong C
- 6.2. Khai báo và sử dụng hàm
- 6.3. Các cách truyền tham số trong hàm
- 6.4. Phân loại biến sử dụng trong chương trình
- 6.5. Nguyên mẫu hàm trong C
- 6.6. Hàm đệ quy

6.4. NGUYÊN MẪU HÀM TRONG C

- Khai báo và định nghĩa hàm
  - Các ví dụ trước tạo và gọi một hàm theo cách sau

```c
#include <stdio.h>

// Create a function
void myFunction() {
  printf("I just got executed!");
}

int main() {
  myFunction(); // call the function
  return 0;
}
```

```
I just got executed!
```

6.4. NGUYÊN MẪU HÀM TRONG C

- Khai báo và định nghĩa hàm
  - Một hàm bao gồm hai phần:
    - Khai báo (**Declaration**): tên hàm, kiểu trả về và các tham số (nếu có)
    - Định nghĩa (**Definition**): phần thân của hàm (mã lệnh được thực thi)

```
void myFunction() { // declaration
  // the body of the function (definition)
}
```

6.4. NGUYÊN MẪU HÀM TRONG C

- Để tối ưu hóa mã, nên tách biệt phần khai báo và phần định nghĩa của hàm.
- Các chương trình C thường có khai báo hàm ***bên trên*** hàm **main** () và định nghĩa hàm ***bên dưới*** hàm **main** → mã lệnh được tổ chức tốt hơn và dễ đọc hơn

khai báo hàm ***bên trên*** hàm **main**() → **nguyên mẫu hàm**

6.4. NGUYÊN MẪU HÀM TRONG C

- Ví dụ:

```c
#include <stdio.h>

// Function declaration
void myFunction();

// The main method
int main() {
  myFunction();  // call the function
  return 0;
}

// Function definition
void myFunction() {
  printf("I just got executed!");
}
```

```
I just got executed!
```

6.4. NGUYÊN MẪU HÀM TRONG C

- Ví dụ:

```c
#include <stdio.h>

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

int main() {
  int result = myFunction(5, 3);
  printf("Result is = %d", result);

  return 0;
}
```

```
Result is = 8
```

6.4. NGUYÊN MẪU HÀM TRONG C

- Ví dụ:

```
#include <stdio.h>
int max(int, int, int); // khai báo nguyên mẫu hàm
void main()
{
  int a,b,c;
  printf("\n Nhap gia tri cho 3 so nguyen a, b, c: ");
  scanf("%d %d %d",&a,&b,&c);
  printf("\n Gia tri cac so vua nhap: ");
  printf(" a = %-5d b = %-5d c = %-5d", a, b, c);
  printf("\n Gia tri lon nhat trong 3 so la %d",max(a,b,c));
}
int max(int x, int y, int z)
{
  int max;
  max = x > y ? x:y;
  max = max > z ? max : z;
  return max;
}
```

```
Nhap gia tri cho 3 so nguyen a, b, c: 45 54 13

Gia tri cac so vua nhap:  a = 45    b = 54    c = 13
Gia tri lon nhat trong 3 so la 54
```

6.4. NGUYÊN MẪU HÀM TRONG C

❖ Ví dụ:

```c
#include <stdio.h>

// Function declaration
int myFunction(int, int);

// The main method
int main() {
  int result = myFunction(5, 3);  // call the function
  printf("Result is = %d", result);
  return 0;
}

// Function definition
int myFunction(int x, int y) {
  return x + y;
}
```

```
Result is = 8
```

6.4. NGUYÊN MẪU HÀM TRONG C

- Nếu muốn đặt phần định nghĩa hàm nằm sau hàm **main()** → Cần khai báo nguyên mẫu của hàm
  - Để báo cho chương trình dịch biết có một hàm có dòng đầu hàm giống như trong phần nguyên mẫu này.
  - Chương trình dịch có thể kiểm tra được là các lời gọi hàm trong chương trình chính có đúng hay không
    - Có phù hợp về kiểu dữ liệu trả về hay không
    - Các tham số thực có kiểu dữ liệu có phù hợp với kiểu dữ liệu đã khai báo hay không.
- Trong hàm nguyên mẫu có thể không cần nêu tên các tham số hình thức, nhưng trong phần khai báo hàm ta cần phải có các tham số hình thức.

6.4. NGUYÊN MẪU HÀM TRONG C

- Ví dụ: Tính tổ hợp chập k của n

```c
#include <stdio.h>
#include <conio.h>
long int TinhGiaiThua(int N); //Nguyên mẫu hàm
int main()
{
  int n,k;
  printf("Nhap vao n:");
  scanf("%d",&n);
  printf("\nNhap vao k:");
  scanf("%d",&k);
  printf("\nTo hop chap k cua n la %ld", TinhGiaiThua(n)/ (TinhGiaiThua(k))* TinhGiaiThua(n-k));
}

long int TinhGiaiThua(int N)
{
  int i;
  long int GT;
  GT=1;
  if (N==0 || N==1) GT=1 ;
  else
    for(i=1; i<=N; i++)   GT = GT*i;
  return GT;
}
```

CHƯƠNG 6: HÀM TRONG C

- 6.1. Khái niệm hàm trong C
- 6.2. Khai báo và sử dụng hàm
- 6.3. Các cách truyền tham số trong hàm
- 6.4. Phân loại biến sử dụng trong chương trình
- 6.5. Nguyên mẫu hàm trong C
- 6.6. Hàm đệ quy

6.6. HÀM ĐỆ QUY

- 6.6.1. Khái niệm đệ quy
- 6.6.2. Xây dựng hàm đệ quy
- 6.6.3. Bài toán áp dụng

6.6.1. KHÁI NIỆM ĐỆ QUY

- Ví dụ 1: Đặt hai chiếc gương cầu đối diện nhau. Trong chiếc gương thứ nhất chứa hình chiếc gương thứ hai. Chiếc gương thứ hai lại chứa hình chiếc gương thứ nhất nên tất nhiên nó chứa lại hình ảnh của chính nó trong chiếc gương thứ nhất…
  - → Ở một góc nhìn hợp lý, ta có thể thấy một dãy ảnh vô hạn của cả hai chiếc gương
- Ví dụ 2:
  - Giai thừa của n (n!): Nếu n = 0 thì n! = 1; nếu n > 0 thì n! = n.(n-1)!

6.6.1. KHÁI NIỆM ĐỆ QUY

- Giải thuật đệ quy
  - Nếu lời giải của một bài toán P được thực hiện bằng lời giải của bài toán P' có dạng giống như P thì đó là một lời giải đệ quy. Giải thuật tương ứng với lời giải như vậy gọi là giải thuật đệ quy.
  - Lưu ý: P' tuy có dạng giống như P, nhưng theo một nghĩa nào đó, nó phải "nhỏ" hơn P, dễ giải hơn P và việc giải nó không cần dùng đến P.

```
3! = 3 * 2!
        ↓
     2! = 2 * 1!
             ↓
          1! = 1 * 0!
                  ↓
               0! = 1
```

6.6.1. KHÁI NIỆM ĐỆ QUY

- Ngôn ngữ C cho phép
  - Gọi từ hàm này tới hàm khác
  - Từ một vị trí trong thân của một hàm gọi tới chính hàm đó → **hàm đệ quy**
- Khi hàm gọi đệ quy đến chính nó thì mỗi lần gọi, máy sẽ tạo ra một tập các **biến cục bộ** mới hoàn toàn **độc lập** với tập các biến cục bộ đã được tạo ra trong các lần gọi trước.
- Có bao nhiêu lần gọi tới hàm thì cũng có bấy nhiêu lần thoát ra khỏi hàm
- Sự tương ứng giữa các lần gọi tới hàm và các lần ra khỏi hàm được thực hiện theo thứ tự ngược, nghĩa là: ***lần ra đầu tiên*** *ứng với* ***lần gọi cuối cùng***

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- **Bài toán nào dùng đệ quy?** → Phương pháp dùng đệ quy thường sử dụng cho các bài toán phụ thuộc tham số có 2 đặc điểm:
  - Bài toán dễ dàng giải quyết trong một số trường hợp riêng ứng với các giá trị đặc biệt của tham số. Đây là **trường hợp suy biến**.
  - Trong **trường hợp tổng quát**, bài toán có thể quy về một bài toán cùng dạng nhưng giá trị tham số bị thay đổi. Và sau một số hữu hạn bước biến đổi đệ quy, sẽ dẫn tới trường hợp suy biến.

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- **Cách viết hàm đệ quy:** 2 cách

```
if (trường hợp suy biến)
  {
     //trình bày cách giải bài toán
  }
else     /*trường hợp tổng quát*/
  {
     //gọi đệ quy tới hàm (đang lập) với giá trị khác của tham số
  }
```

```
if /*trường hợp tổng quát*/
  {
        //gọi đệ quy tới hàm (đang lập) với giá trị khác của tham số
  }
else      (trường hợp suy biến)
  {
        //trình bày cách giải bài toán

  }
```

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- Ví dụ:
  - Giải thích: Khi hàm **sum** () được gọi, nó sẽ thêm tham số k vào tổng của tất cả các số nhỏ hơn k và trả về kết quả. Khi k trở thành 0, hàm chỉ trả về 0. Khi chạy, chương trình thực hiện theo các bước:

```
10 + sum(9)
10 + ( 9 + sum(8) )
10 + ( 9 + ( 8 + sum(7) ) )
...
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + sum(0)
10 + 9 + 8 + 7 + 6 + 5 + 4 + 3 + 2 + 1 + 0
```

```c
#include <stdio.h>

int sum(int k);

int main() {
  int result = sum(10);
  printf("%d", result);
  return 0;
}

int sum(int k) {
  if (k > 0) {
    return k + sum(k - 1);
  } else {
    return 0;
  }
}
```

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- Ví dụ 1:
  - Tính tổng 1+2+…+n sử dụng đệ quy

```
#include<stdio.h>
int calculateSum(int);
int main()
{
  int i, num, result;
  printf("Nhap mot so bat ky: "); scanf("%d", &num);
  result = calculateSum(num);
  printf("\nTong cac so tu 1 toi %d la: %d", num, result);
  return (0);
}
int calculateSum(int num) {
  int res;
  if (num == 1) {
    return (1);
  } else
  {
    res = num + calculateSum(num - 1);
  }
  return (res);
}
```

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- Ví dụ 2: In dãy Fibonacci sử dụng đệ quy
  - Dãy số Fibonacci bắt nguồn từ bài toán cổ về việc sinh sản của các cặp thỏ:
    1. Các con thỏ không bao giờ chết
    2. Hai tháng sau khi ra đời, mỗi cặp thỏ mới sẽ sinh ra một cặp thỏ con (một đực, một cái)
    3. Khi đã sinh con rồi thì cứ mỗi tháng tiếp theo chúng lại sinh được một cặp con mới
    4. Giả sử từ đầu tháng 1 có một cặp mới ra đời thì đến giữa tháng thứ n sẽ có bao nhiêu cặp.

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- Ví dụ 2: In dãy Fibonacci sử dụng đệ quy
  - Ví dụ, n = 5, ta thấy:
    - Giữa tháng thứ 1: 1 cặp (ab) (cặp ban đầu)
    - Giữa tháng thứ 2: 1 cặp (ab) (cặp ban đầu vẫn chưa đẻ)
    - Giữa tháng thứ 3: 2 cặp (AB)(cd) (cặp ban đầu đẻ ra thêm 1 cặp con)
    - Giữa tháng thứ 4: 3 cặp (AB)(cd)(ef) (cặp ban đầu tiếp tục đẻ)
    - Giữa tháng thứ 5: 5 cặp (AB)(CD)(ef)(gh)(ik) (cả cặp (AB) và (CD) cùng đẻ)
    - → tính số cặp thỏ ở tháng thứ n: **F(n)**

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- Ví dụ 2: In dãy Fibonacci sử dụng đệ quy
  - Tính số cặp thỏ ở tháng thứ n: **F(n)**
    - Nếu mỗi cặp thỏ ở tháng thứ n - 1 đều sinh ra một cặp thỏ con thì số cặp thỏ ở tháng thứ n sẽ là: F(n) = 2 * F(n - 1)
    - Nhưng vấn đề không phải như vậy, trong các cặp thỏ ở tháng thứ n - 1, chỉ có những cặp thỏ đã có ở tháng thứ n - 2 mới sinh con ở tháng thứ n được thôi.
    - Do đó F(n) = F(n - 1) + F(n - 2) (= số cũ + số sinh ra).
    - Vậy có thể tính được F(n) theo công thức sau:

      F(n) = 1 nếu n ≤ 2

      F(n) = F(n - 1) + F(n - 2) nếu n > 2

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- Ví dụ 2: In dãy Fibonacci sử dụng đệ quy
  - Dãy Fibonacci là dãy số được tạo bằng cách: số kế tiếp bằng tổng của hai số liền trước.
  - Dãy Fibonacci bắt đầu từ hai số F0 & F1. Giá trị ban đầu của F0 & F1 có thể tương ứng là 0, 1 hoặc 1, 1.

→Điều kiện của dãy Fibonacci có thể tổng quát:

ví dụ hai dãy Fibonacci
F_8 = 0 1 1 2 3 5 8 13 hoặc:
F_8 = 1 1 2 3 5 8 13 21

$$F_n = F_{n-1} + F_{n-2}$$

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- Ví dụ 2: In dãy Fibonacci

```c
#include <stdio.h>
int sofibonaci(int i)
{
    if(i <= 2)
        return 1;
    return sofibonaci(i-1) + sofibonaci(i-2);
}
int main()
{
    int i, n;
     printf("Nhanp n= "); scanf("%d", &n);
    for (i = 1; i <= n; i++)
        printf("%d\t", sofibonaci(i));
    return 0;
}
```

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

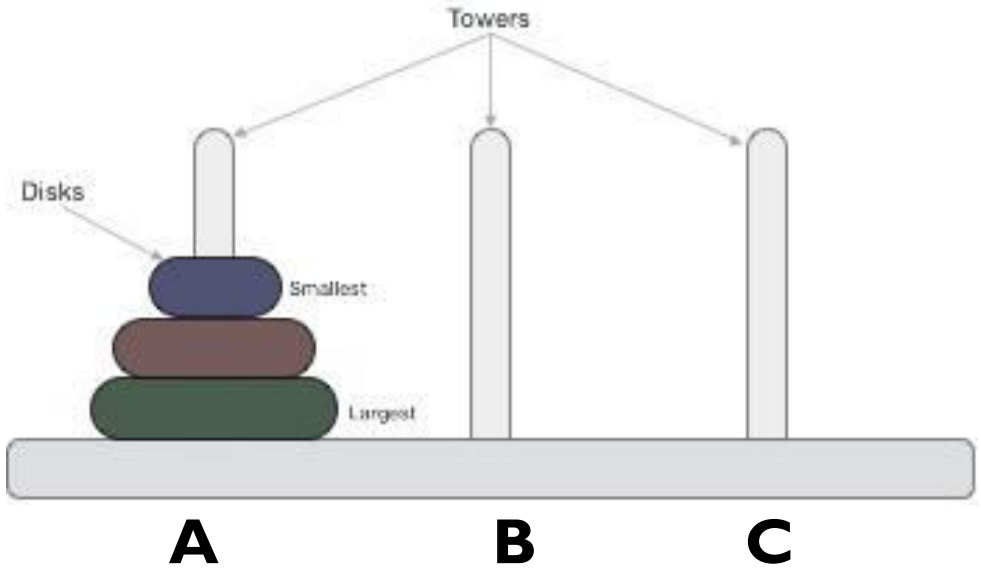
- Ví dụ 3: Tính giai thừa sử dụng đệ quy

```
# include <stdio.h>
float giaithua(int a)
{
    if (a<=1)
        return 1;
    else
        return (a*giaithua(a-1));
}

void main()
{
    int n;
    printf("Nhap n= "); scanf("%d",&n);
    printf("%d!= %.0f", n, giaithua(n));
}
```

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- Ví dụ 4: Giải bài toán Tháp Hà Nội (Tower of Hanoi) sử dụng đệ quy
  - Bài toán Tháp Hà Nội (Tower of Hanoi) là một trò chơi toán học bao gồm 3 cột và với số đĩa nhiều hơn 1.
  - Minh họa bài toán Tháp Hà Nội (Tower of Hanoi) với trường hợp có 3 đĩa:



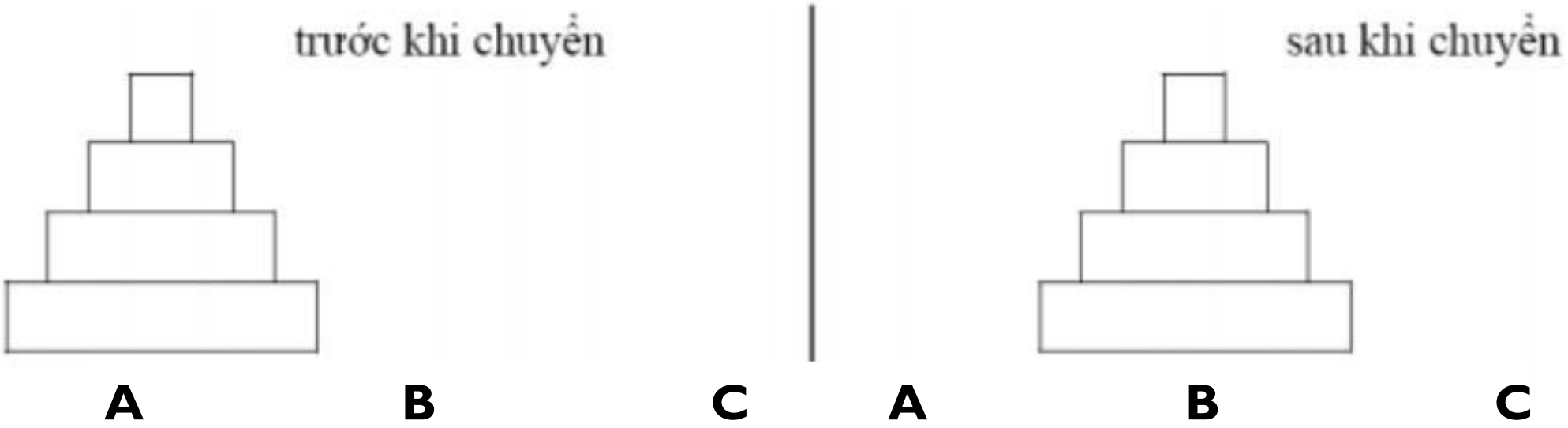
Các đĩa có kích cỡ khác nhau và xếp theo tự tự tăng dần về kích cỡ từ trên xuống: đĩa nhỏ hơn ở trên đĩa lớn hơn.

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- Ví dụ 4: Bài toán Tháp Hà Nội (Tower of Hanoi)
  - Có n đĩa với kích thước to, nhỏ khác nhau. Ban đầu sắp xếp các đĩa theo trật tự kích thước vào một cọc sao cho đĩa nhỏ nhất nằm trên cùng. Người chơi phải di chuyển toàn bộ số đĩa sang một cọc khác, tuân theo các quy tắc sau:
    - Khi di chuyển một đĩa, phải đặt nó vào một trong ba cọc đã cho
    - Mỗi lần chỉ có thể chuyển một đĩa và phải là đĩa ở trên cùng
    - Tại một vị trí, đĩa nào mới chuyển đến sẽ phải đặt lên trên cùng
    - Đĩa lớn hơn không bao giờ được phép đặt lên trên đĩa nhỏ hơn

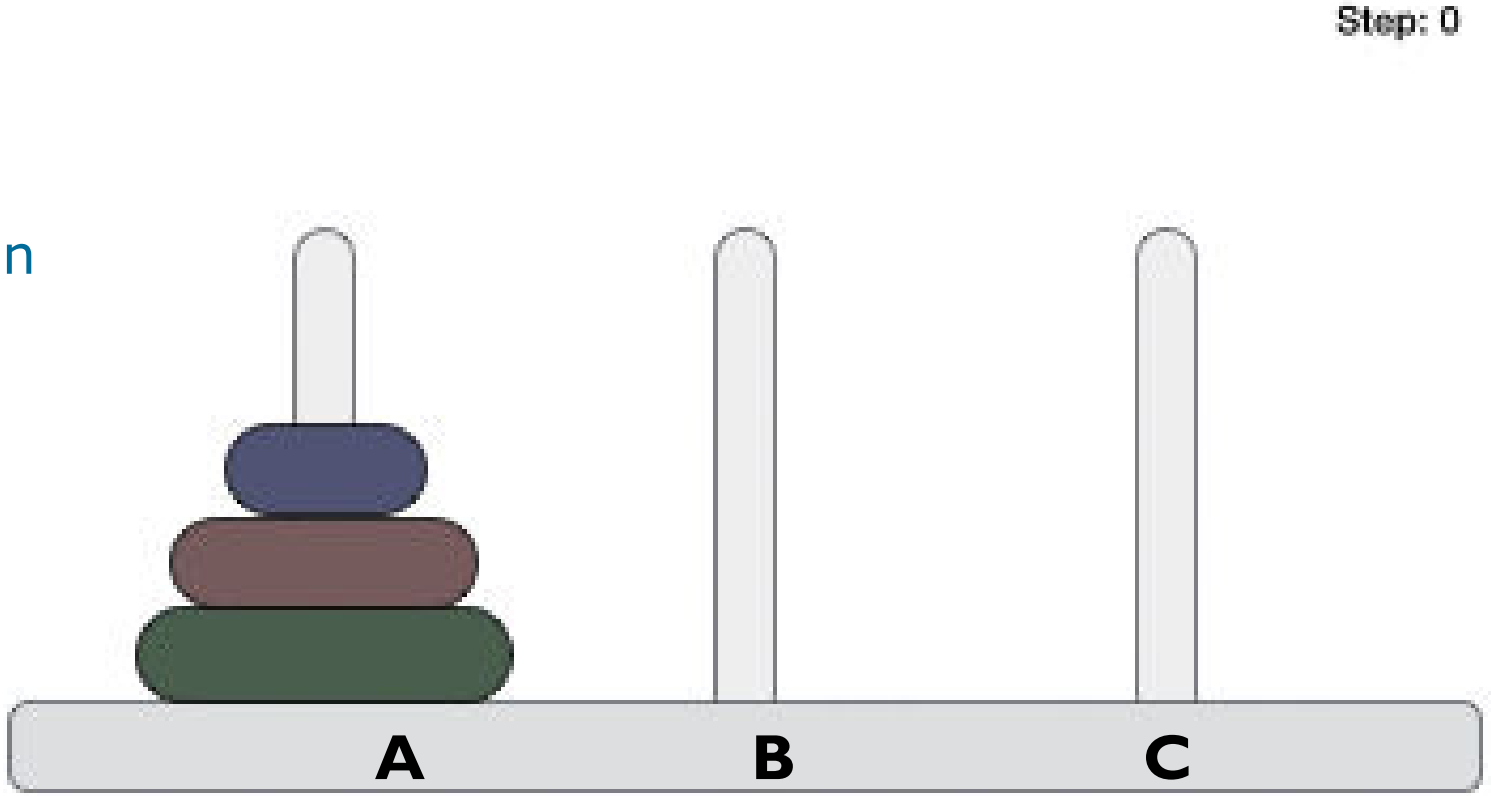
6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- Ví dụ 4: Bài toán Tháp Hà Nội (Tower of Hanoi)
  - Hình ảnh mô tả bài toán:



6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- Ví dụ 4: Bài toán Tháp Hà Nội (Tower of Hanoi)
  - Minh họa cách giải bài toán với trường hợp có 3 đĩa.



Với số đĩa là n có thể được giải với số bước tối thiểu là 2^n-1
→ Với trường hợp 3 đĩa, bài toán có thể được giải sau $2^3-1 = 7$ bước

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

- Ví dụ 4: Bài toán Tháp Hà Nội (Tower of Hanoi)
  - Mô tả bài toán:
    - Đặt tên các cọc là A, B, C và qui ước A là cọc nguồn, B là cọc đích và C là cọc trung gian;
    - Gọi n là tổng số đĩa;
    - Đánh số đĩa từ 1 (nhỏ nhất, trên cùng) đến n (lớn nhất, dưới cùng).
    - Thuật giải đệ qui: Để chuyển n đĩa từ cọc A sang cọc B với C là cọc trung gian:

      1. Chuyển n-1 đĩa từ A sang C. Chỉ còn lại đĩa thứ n trên cọc A;

      2. Chuyển đĩa thứ n từ A sang B;

      3. Chuyển n-1 đĩa từ C sang B cho chúng nằm trên đĩa thứ n

6.6.2. XÂY DỰNG HÀM ĐỆ QUY

```
Cot nguon: A     cot dich: B     cot trung gian: C
Nhap so dia n= 1
Chuyen dia 1 tu A -> B
```

```
Cot nguon: A     cot dich: B     cot trung gian: C
Nhap so dia n= 2
Chuyen dia 1 tu A -> C
Chuyen dia 2 tu A -> B
Chuyen dia 1 tu C -> B
```

```
Cot nguon: A     cot dich: B     cot trung gian: C
Nhap so dia n= 3
Chuyen dia 1 tu A -> B
Chuyen dia 2 tu A -> C
Chuyen dia 1 tu B -> C
Chuyen dia 3 tu A -> B
Chuyen dia 1 tu C -> A
Chuyen dia 2 tu C -> B
Chuyen dia 1 tu A -> B
```

```c
#include <stdio.h>
void chuyen(int sodia, char cotNguon, char cotDich, char cotTG)
{
    if (sodia > 0)
    {
        chuyen(sodia-1, cotNguon, cotTG, cotDich);
        printf("Chuyen dia %d tu %c -> %c\n", sodia, cotNguon, cotDich);
        chuyen(sodia-1, cotTG, cotDich, cotNguon);
    }
}
int main()
{
    char cotNguon, cotDich, cotTG;
    int n;
    printf("Cot nguon: A\t cot dich: B\t cot trung gian: C");
    printf("\nNhap so dia n= "); scanf("%d", &n);
    chuyen(n,'A', 'B', 'C');
}
```

TÓM TẮT CHƯƠNG 6

- Xây dựng được hàm:
  - Khai báo nguyên mẫu hàm
  - Định nghĩa hàm
  - Gọi hàm
  - Truyền tham số cho hàm
- Khái niệm hàm đệ quy, các trường hợp áp dụng hàm đệ quy, cách xây dựng hàm đệ quy.
- Vận dụng hàm để giải các bài toán đệ quy

VÍ DỤ

- Ví dụ 1. Viết hàm tính giá trị trung bình của ba số nguyên.
- Ví dụ 2. Viết hàm đọc vào 1 số nguyên dương, khi đọc vào số âm, hàm sẽ yêu cầu đọc lại.
- Ví dụ 3. Viết hàm tính S=1!+2!+…+n!

BÀI TẬP 1

- Viết hàm đếm số từ của một xâu ký tự
  - Yêu cầu: Viết chương trình cho phép nhập vào một xâu ký tự, đếm số từ có trong xâu. Từ được hiểu là một đoạn ký tự nằm giữa hai kí tự cách (space) và khác kí tự cách.

BÀI TẬP 1

- Đếm số từ

```c
#include <stdio.h>
int demTu(char *s);
int main()
{
    char s[100];
    printf("Nhap vao 1 chuoi: ");
    fflush(stdin); gets(s);
    printf("So tu: %d", demTu(s));
    return 0;
}
int demTu(char *s)
{
    int dem = 0, i = 0;
    while (s[i] != '\0')
    {
        if (s[i] != ' ' && (s[i+1] == ' ' || s[i+1] == '\0')) dem = dem + 1;
        i++;
    }
    return dem;
}
```

```
Nhap vao 1 chuoi: cong nghe   thong tin
So tu: 4
```

BÀI TẬP

- Bài tập 2:
  - Viết 1 hàm cho phép nhập vào số nguyên, 1 hàm cho phép nhập vào số thực.
- Bài tập 3:
  - Viết chương trình cho phép nhập vào tọa độ trên mặt phẳng XOY ba đỉnh của tam giác ABC, và tọa độ của điểm M; đưa ra kết luận về vị trí tương đối của M so với ABC: M nằm trong, trên cạnh hay nằm ngoài ABC.
- Bài tập 4:
  - Cho tam giác ABC và một điểm M không nằm trên cạnh của tam giác. Hãy viết chương trình tính số lớn nhất R sao cho vòng tròn tâm M, bán kính R, không cắt bất cứ cạnh nào và không chứa tam giác.

BÀI TẬP

- Bài tập 5:
  - Viết hàm chuẩn hóa chuỗi họ tên người Việt theo yêu cầu sau: Không có ký tự trống phía đầu và cuối, giữa 2 từ có một ký tự trống, ký tự đầu tiên của mỗi từ được viết HOA và ký tự khác được viết thường.
- Bài tập 6:
  - Giả sử hovaten là xâu họ tên người việt, viết các hàm TEN cho phép lấy ra phần tên, hàm HODEM cho phép lấy ra phần họ và đệm của hovaten.
- Bài tập 7:
  - Viết hàm đếm số từ của một xâu ký tự. Mỗi từ được hiểu là đoạn ký tự giữa hai dấu trống và không phải là khoảng trống.

BÀI TẬP

- Bài tập 8:
  - Viết hàm chuyển số nguyên thành xâu ký tự biểu diễn cách đọc số đó. Ví dụ nhập vào số:
    - 235 → in ra "Hai tram ba muoi lam"
    - 807 → in ra "Tam tram linh bay"
    - 55 → in ra "Nam muoi lam".

BÀI TẬP

- Bài tập 9:
  - Xâu ký tự biểu diễn ngày tháng là xâu dạng "DD/MM/YYY", trong đó DD là hai chữ số biểu diễn ngày, MM là hai chữ số biểu diễn tháng và YYYY là bốn chữ số biểu diễn năm. Viết hàm kiểm tra một xâu ký tự nhập vào có phải là xâu biểu diễn ngày tháng hay không? Nếu là xâu biểu diễn ngày tháng thì ngày tháng đó có hợp lệ hay không.
    - Ví dụ xâu '30/02/2010' là xâu biểu diễn ngày tháng nhưng ngày tháng đó là không hợp lệ (vì không có ngày 30 tháng 2).

BÀI TẬP

- Bài tập 10:
  - Cho hai xâu biểu diễn ngày tháng **sd1** và **sd2**, viết hàm tính số ngày tính từ **sd1** đến **sd2** nếu **sd1** và **sd2** là hai xâu biểu diễn ngày tháng hợp lệ (tham khảo bài 9).
- Bài tập 11:
  - Cho **sd1** là xâu biểu diễn ngày tháng, n là một số nguyên.
  - Viết hàm **NgaySauN(sd1,n)** trả về xâu ký tự dạng ngày tháng là ngày thứ n sau ngày **sd1**.
  - Viết hàm **NgayTruocN(sd1,n)** trả về xâu ký tự dạng ngày tháng là ngày thứ n trước ngày **sd1**
    - Ví dụ **sd1** = "27/01/2010" và n= 46 thì **NgayTruocN(sd1,n)** trả về "12/12/2009".
    - Ví dụ **sd1** = "27/01/2010" và n= 15 thì **NgaySauN(sd1,n)** trả về "11/02/2010".

BÀI TẬP

- Bài tập 12:
  - Viết hàm cho phép đổi năm dương lịch thành năm âm lịch. Ví dụ nếu năm dương lịch là 2010 thì năm âm lịch là "Canh Dần".
- Bài tập 13:
  - Ta gọi tổng triệt để các chữ số của số tự nhiên n là tổng các chữ số của n nếu tổng này chỉ có một chữ số; trong trường hợp ngược lại tổng triệt để là kết quả của việc cộng dồn liên tiếp cho đến khi chỉ còn một chữ số.
    - Ví dụ số n=1186 có tổng các chữ số là 16, số này có hai chữ số và tổng các chữ số của nó là 7. Vậy tổng triệt để của n=1186 là 7.
  - Nhập từ bàn phím một số k, $1 \le k \le 9$, viết ra màn hình tất cả các số có 4 chữ số mà tổng triệt để của nó bằng k (không kể các hoán vị).

BÀI TẬP

- Bài tập 14:
  - Một sân hình chữ nhật được chia thành m×n ô vuông có kích thước bằng nhau, với m, n≤50. Mỗi ô có thể được chôn một quả mìn. Nếu nổ một quả mìn ở ô (i,j), tức là ở hàng i và cột j, thì nó sẽ gây nổ cho các ô xung quanh (i+1,j), (i-1,j), (i,j+1), (i,j-1) nếu các ô này có mìn. Hãy nhập vào vị trí các ô có mìn và vị trí ô đầu tiên bị nổ và in ra tối đa các ô bị nổ trên sân chữ nhật.

BÀI TẬP

- Bài tập 15:
  - Cho một dãy có n+1 ô được đánh số từ 0 đến n. Một quân cờ đứng ở ô số 0. Mỗi **một nước đi** quân cờ được đi lên phía trước không quá k ô. Một **cách đi** của quân cờ là xuất phát từ ô số 0 quân cờ phải đi đến ô thứ n. Với hai số nguyên dương n và k, hãy tính số cách đi của quân cờ. Ví dụ, với n=3 và k=2 sẽ có 3 cách đi quân cờ với các nước 1+1+1, 1+2 và 2+1.
    - (Bài 1, Đề thi Olympic Tin học Sinh viên Đại học Tổng hợp Hà nội mở rộng, Khối không chuyên, Hà nội 4-1996).

BÀI TẬP

- Bài tập 16:
  - Xây dựng hàm tính giai thừa cách của một số nguyên

    n!! = 1x3x5x … x n nếu n lẻ

    n!! = 2x4x6x … x n nếu n chẵn
- Bài tập 17:
  - Viết một hàm tính tổng các chữ số của một số nguyên.
- Bài tập 18:
  - Tổ hợp chập k từ n, kí hiệu là C_n^k. Viết chương trình tính C_n^k biết rằng:

    $C_n^k = C_{n-1}^{k-1} + C_{n-1}^k$ và $C_n^0 = C_n^n = C_k^k = 1$
- Bài tập 19:
  - Viết ra tất cả các hoán vị của dãy X_n={1,2,...,n} có n phần tử là các số 1,2 ,..., n.

TO FINISH

THANKS.

FOR YOUR ATTENTION