ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG

TIN HỌC ĐẠI CƯƠNG

Phần II: LẬP TRÌNH C

Nội dung chính

- Chương 1: Tổng quan về ngôn ngữ C
- Chương 2: Kiểu dữ liệu và biểu thức trong C
- Chương 3: Vào ra dữ liệu
- Chương 4: Cấu trúc điều khiển
- Chương 5: Mảng, con trỏ và xâu kí tự
- Chương 6: Cấu trúc
- Chương 7: Hàm
- Chương 8: Tệp dữ liệu

Chương 7: Hàm

7.1. Khái niệm hàm

- Khái niệm chương trình con
- Chương trình con trong ngôn ngữ C

7.2. Khai báo và sử dụng hàm

- Khai báo và sử dụng

7.3. Phạm vi của biến

- Toàn cục và địa phương
- Biến static, biến register

7.4. Truyền tham số

- Truyền theo giá trị, truyền theo địa chỉ

Chương 7: Hàm

7.1. Khái niệm hàm

- Khái niệm chương trình con
- Chương trình con trong ngôn ngữ C

7.2. Khai báo và sử dụng hàm

- Khai báo và sử dụng

7.3. Phạm vi của biến

- Toàn cục và địa phương
- Biến static, biến register

7.4. Truyền tham số

- Truyền theo giá trị, truyền theo địa chỉ

Khái niệm chương trình con

- Khái niệm
  - Là đoạn chương trình thực hiện một nhiệm vụ cụ thể và được đóng gói theo cấu trúc xác định
  - Được định nghĩa một lần và sử dụng lại nhiều lần.
- Vai trò
  - Tổ chức một chương trình thành nhiều phần nhỏ để tăng tính cấu trúc.
  - Tiết kiệm thời gian bằng cách sử dụng chương trình con đã có thay vì viết lại.
  - Mở rộng tính năng cho chương trình bằng cách sử dụng các thư viện, ví dụ printf(), scanf()…
  - Gỡ lỗi, bảo trì chương trình dễ dàng hơn

Hàm trong ngôn ngữ C

- Các ngôn ngữ lập trình nói chung: 2 loại chương trình con
  - **Hàm**: thực hiện công việc và trả lại kết quả.
  - **Thủ tục**: thực hiện công việc, không trả lại kết quả.
- Ngôn ngữ C:
  - Chỉ có 1 loại chương trình con là hàm (function).
  - Hàm trong C tương đương cả hàm và thủ tục trong các ngôn ngữ khác.
  - Sử dụng kiểu **void** (kiểu dữ liệu không định kiểu) khi hàm không trả về dữ liệu.

Chương 7: Hàm

7.1. Khái niệm hàm
- Khái niệm chương trình con
- Chương trình con trong ngôn ngữ C

7.2. Khai báo và sử dụng hàm
- Khai báo và sử dụng

7.3. Phạm vi của biến
- Toàn cục và địa phương
- Biến static, biến register

7.4. Truyền tham số
- Truyền theo giá trị, truyền theo địa chỉ

Ví dụ

Khai báo chương trình con

Gọi chương trình con ra thực hiện

```
#include<stdio.h>
int bp(int x) {
    int y;
    y =  x * x;
    return y;
}
int main() {
   int i;
   for (i=1; i<20; i+=2)
      printf("%4d\n", bp(i));
   printf("\n");
   return 0;
}
```

```
   1
   9
  25
  49
  81
 121
 169
 225
 289
 361
```

Định nghĩa hàm

Cú pháp

Dòng đầu hàm

```
Kiểu_hàm Tên_hàm(DS khai báo tham số)
{
  [<Các khai báo cục bộ>]
  [<Các câu lệnh>]
}
```

Thân hàm

Dòng đầu hàm

Kiểu_hàm Tên_hàm(DS khai báo tham số)

- Mô tả các thông tin được trao đổi giữa bên trong và bên ngoài hàm.
  - Tên của hàm,
  - Các tham số đầu vào
    - Hàm cần những thông tin gì để hoạt động
  - Tham số đầu ra và giá trị trả về
    - Hàm cung cấp những thông tin gì cho môi trường
- Dùng phân biệt các hàm với nhau,
  - không tồn tại 2 hàm có dòng đầu hàm giống nhau.

Dòng đầu hàm→Tên hàm

Là tên do người sử dụng tự định nghĩa

- Tuân theo quy tắc đặt tên đối tượng
- Nên mang ý nghĩa gợi ý chức năng của hàm

Dòng đầu hàm→Khai báo các tham số hình thức

- Khai báo các thông tin cần cho hoạt động của hàm và các thông tin, kết quả tính toán được hàm trả lại.
  - Tham số chứa dữ liệu vào cung cấp cho hàm
  - Tham số chứa dữ liệu ra mà hàm tính toán được.
- Các tham số sử dụng trong khai báo hàm là tham số hình thức.
  - Nguyên tắc khai báo tham số hình thức như giống như khai báo một biến

    kiểu_dữ_liệu_của_tham_số *tên_của_tham_số*

Dòng đầu hàm→Khai báo các tham số hình thức

- Các tham số cung cấp cho hàm trong quá trình thực hiện hàm là tham số thực sự
  - Kiểu dữ liệu của tham số thực phải giống kiểu dữ liệu của tham số hình thức tương ứng với tham số thực đó.
- Một hàm có thể có một, nhiều hoặc không có tham số nào cả
  - Nếu có nhiều tham số, phải được phân cách với nhau bằng dấu phẩy.
  - Nếu không có tham số vẫn phải có cặp dấu ngoặc đơn sau tên hàm.

Dòng đầu hàm→Kiểu dữ liệu trả về

- Thông thường hàm sau khi được thực hiện sẽ trả về một giá trị kết quả tính toán nào đó.
- Để sử dụng được giá trị đó cần phải biết nó thuộc kiểu dữ liệu gì.
  - Kiểu dữ liệu của đối tượng tính toán được hàm trả về được gọi là kiểu dữ liệu trả về của hàm.

Dòng đầu hàm→Kiểu dữ liệu trả về

- 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**.
- Sử dụng kiểu dữ liệu trả về là **void** nếu hàm không trả về giá trị nào cả.
- Nếu không khai báo kiểu dữ liệu trả về thì chương trình dịch của C sẽ ngầm hiểu rằng kiểu dữ liệu trả về của hàm là kiểu **int**.

Thân hàm

- Danh sách các câu lệnh
- Thường có ít nhất một lệnh **return**

Hoạt động của hàm:

- Thực hiện lần lượt các lệnh cho đến khi một trong 2 điều kiện thỏa mãn:
  - Thực hiện xong tất cả các câu lệnh có trong thân hàm.
  - Gặp lệnh **return**
    - Cú pháp chung `return [biểu_thức];`

Thân hàm (tiếp)

Khi gặp lệnh **`return biểu_thức;`**

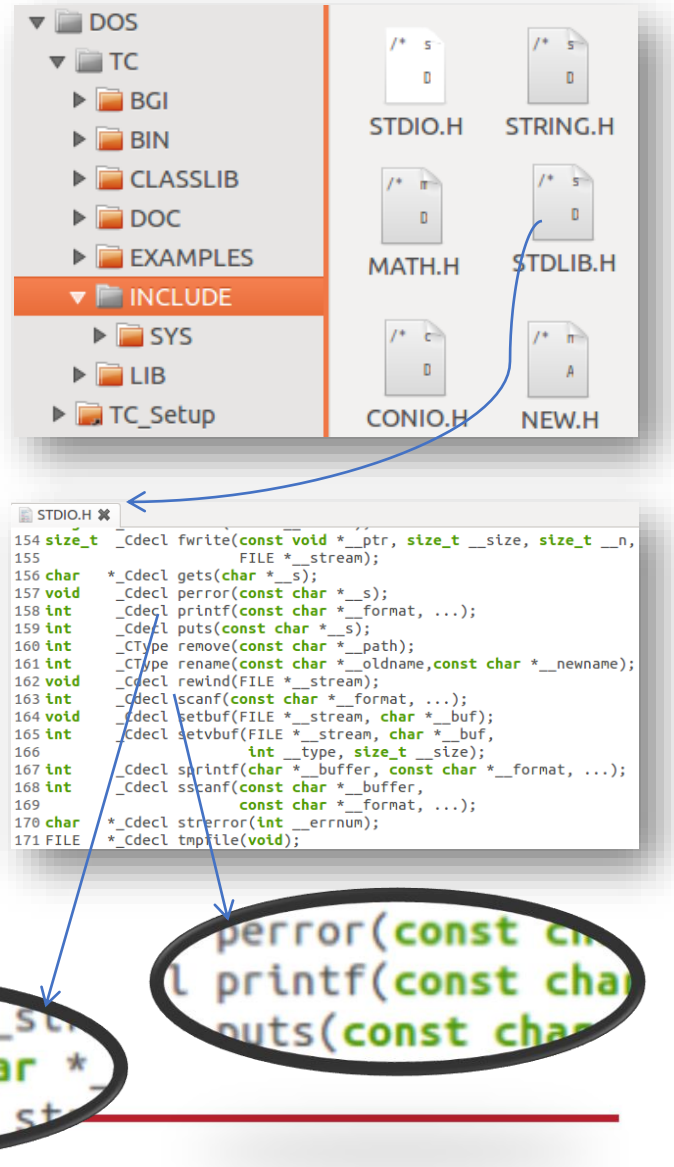
- Tính toán giá trị của *biểu_thức*,
- Lấy kết quả tính toán được làm giá trị trả về cho lời gọi hàm
- Kết thúc việc thực hiện hàm, trở về, thực hiện câu lệnh sau lời gọi hàm.

Nếu **return** không có phần *biểu_thức*,

- Kết thúc thực hiện hàm mà không trả về giá trị nào cả.
  - Dùng khi hàm được khai báo có kiểu trả về là **void**

Các hàm thư viện

- Ngôn ngữ C cung cấp một số hàm thư viện như: xử lý vào ra, hàm toán học, hàm xử lý xâu…
- Để sử dụng các hàm này chúng ta chỉ cần khai báo nguyên mẫu của chúng trước khi sử dụng.
  - Khai báo thông qua chỉ thị **#include** <tệp_tiêu_đề>
  - Tệp_tiêu_đề (.h) đã chứa các nguyên mẫu hàm



Sử dụng hàm

Tên_hàm (DS_tham_số_thực _sự);

Ví dụ:

N = bp(1);N= bp(3);,…

Lưu ý:

- Gọi hàm thông qua tên hàm và các tham số được cung cấp thực sự cho hàm (*tham số thực sự*).
- Nếu hàm nhận nhiều tham số thì các tham số ngăn cách nhau bởi dấu phẩy
- Các tham số hình thức của hàm sẽ nhận các giá trị từ tham số truyền vào
- Sau khi thực hiện xong, trở về điểm mà hàm được gọi

Ví dụ khai báo hàm

- Ví dụ: hàm tính giai thừa

```
int giai_thua(int a)
{
   int ket_qua;
   int i;

   if(a < 0) ket_qua = -1;
   else if(a == 0) ket_qua = 1;
   else {
      ket_qua = 1;
      for (i = 1; i <= a; i++)
         ket_qua = ket_qua * i;
   }
   return ket_qua;
}
```

——————> Dòng đầu hàm

——————> Các khai báo

——————> Các câu lệnh

Ví dụ khai báo hàm

- Ví dụ: hàm tính giai thừa

```
int giai_thua(int a)
{
   int ket_qua;
   int i;

   if(a < 0) return -1;
   if(a == 0) return 1;

   ket_qua = 1;
   for (i = 1; i <= a; i++)
      ket_qua = ket_qua * i;

   return ket_qua;
}
```

——————> Dòng đầu hàm

——————> Các khai báo

——————> Các câu lệnh

Ví dụ khai báo hàm

- Ví dụ: hàm tìm số lớn nhất trong 3 số a, b, c

```
int max(int a, int b, int c)
{
   int ket_qua;

   ket_qua = a;
   if(ket_qua < b) ket_qua = b;
   if(ket_qua < c) ket_qua = c;

   return ket_qua;
}
```

——————> Dòng đầu hàm

——————> Các khai báo

——————> Các câu lệnh

Ví dụ khai báo hàm

- Ví dụ: hàm (thủ tục) vẽ tam giác *

```
void tamgiac(int n)
{
  int i, j, k;

  for (i=1; i<=n; i++)
  {
    for (k=1; k<=n-i; k++)
      printf(" ");
    for (j=1;j<=2*i-1;j++)
      printf("*");
    printf("\n");
  }
}
```

Dòng đầu hàm

Các khai báo

Các câu lệnh

Ví dụ khai báo hàm

- Ví dụ: hàm (thủ tục) vẽ tam giác *

```
void tamgiac()
{
   int i, j, k, n = 10;

   for (i=1; i<=n; i++)
   {
     for (k=1; k<=n-i; k++)
       printf(" ");
     for (j=1;j<=2*i-1;j++)
       printf("*");
     printf("\n");
   }
}
```

——————> Dòng đầu hàm

——————> Các khai báo

——————> Các câu lệnh

Ví dụ

- Cách sử dụng hàm

```
g = giaithua(10);
g = giaithua(n);

m = max(10, 20, 30);
m = max(a, b, c);

tamgiac(10);
tamgiac(n);
tamgiac();
```

Ví dụ

- Chương trình tính số tổ hợp, chỉnh hợp

```
#include <stdio.h>

int GT(int n)
{
	int ket_qua, i;
	if (n < 0) return -1;
	if (n == 0) return 1;

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

	return ket_qua;
}
```

Ví dụ

```
// khai bao ham tinh chinh hop A(k, n)
int A(int k, int n)
{
	return GT(n) / GT(n - k);
}
// khai bao ham tinh top hop C(k, n)
int C(int k, int n)
{
	return GT(n) / (GT(k) * GT(n - k));
}
int main() {
	int n = 4, k = 3;
	printf("%d! = %d\n", n, GT(n));
	printf("%d! = %d\n", k, GT(k));
	printf("A(%d, %d) = %d\n", k, n, A(k, n));
	printf("C(%d, %d) = %d\n", k, n, C(k, n));
	return 0;
}
```

Truyền tham số cho hàm main()

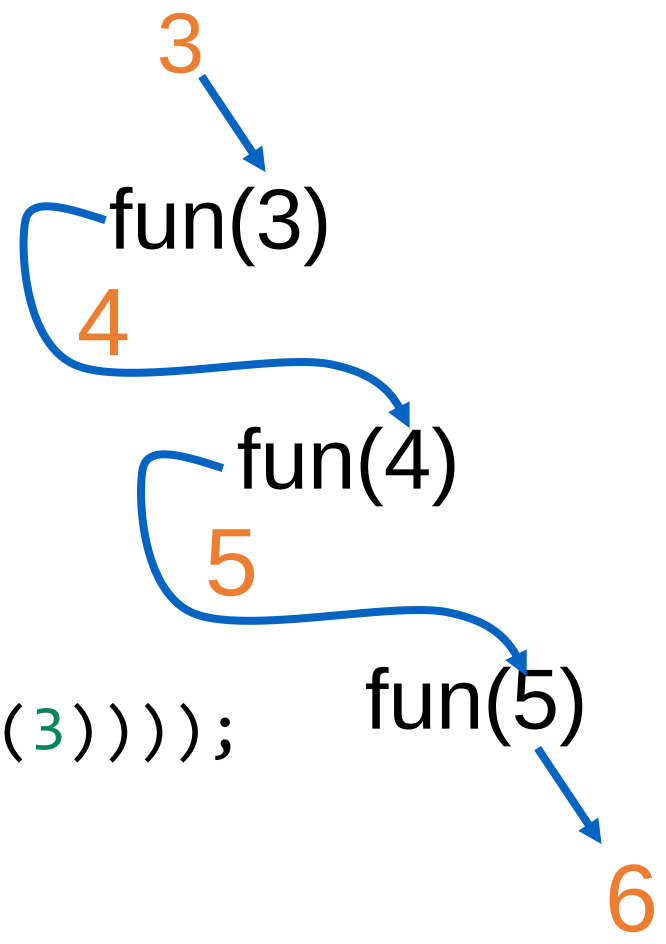
```
#include <stdio.h>

int main(int argc, char *argv[])
{
      int i, j, k, n;
      printf("Danh sach cac tham so: \n");
      for (i = 0; i < argc; i++)
         printf("%s\n", argv[i]);
      n = atoi(argv[1]);
      printf("n = %d\n", n);
```

Truyền tham số cho hàm main()

```
        for (i = 1; i <= n; i++)
        {
                for (k = 1; k <= n - i; k++)
                        printf(" ");
                for (j = 1; j <= 2 * i - 1; j++)
                        printf("*");
                printf("\n");
        }
        return 0;
}
```

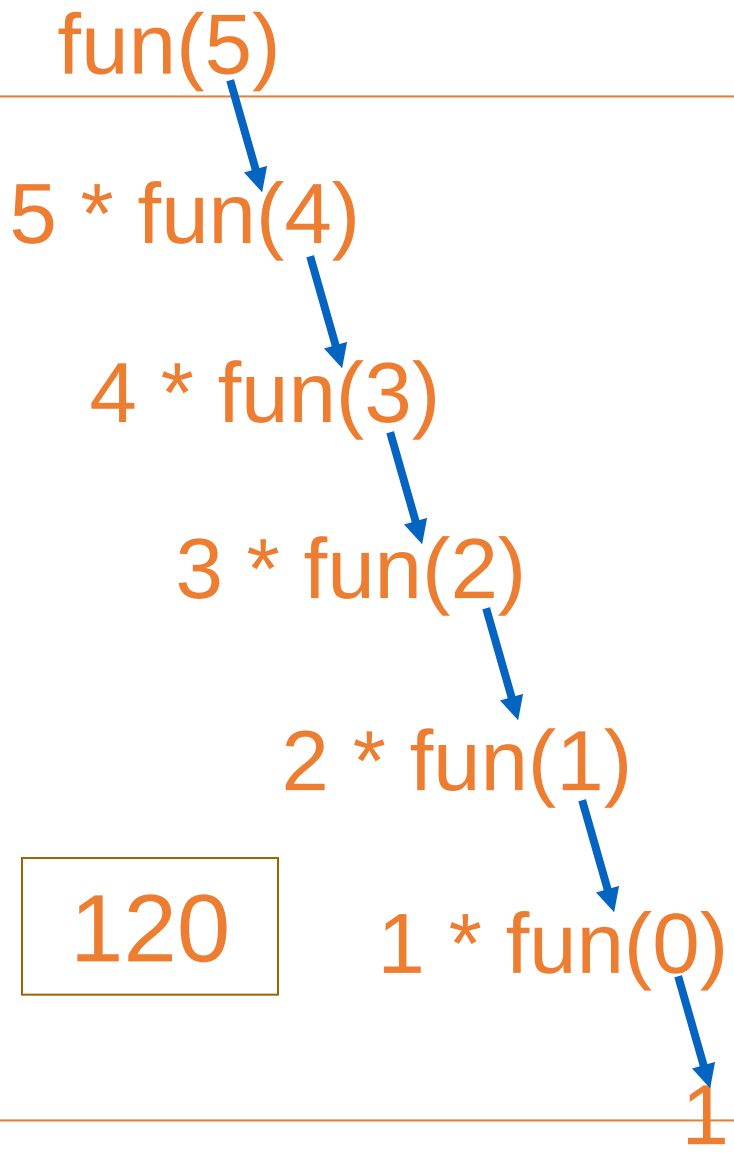
VD1: Cho biết kết quả thực hiện chương trình

```
#include <stdio.h>

int fun(int a) {
    a++;
    return a;
}

int main() {
    printf("%d\n", fun(fun(fun(3))));
    return 0;
}
```



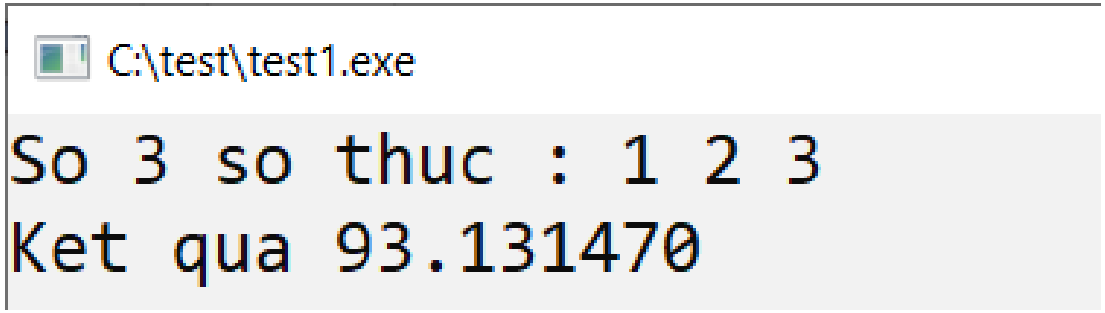
VD2 : Cho biết kết quả thực hiện chương trình

```c
#include <stdio.h>

int fun(int n) {
    if (n == 0)
        return 1;
    else
        return n * fun(n - 1);
}

int main() {
    printf("%d\n", fun(5));
    return 0;
}
```



VD3: Tính TBC f(a),f(b), f(c) nếu $f(x) = x^5 + \sqrt[5]{x}$

```c
#include <stdio.h>
#include <math.h>
float f(float x) {
    if (x == 0.0)
        return 0;
    else
        return pow(x, 5) + x / fabs(x) * pow(fabs(x), 0.2);
}
int main() {
    float a, b, c;
    printf("So 3 so thuc : ");
    scanf("%f%f%f", &a, &b, &c);
    printf("Ket qua %f \n", (f(a) + f(b) + f(c)) / 3);
    return 0;
}
```



VD4: Tìm ƯSCLN của dãy số

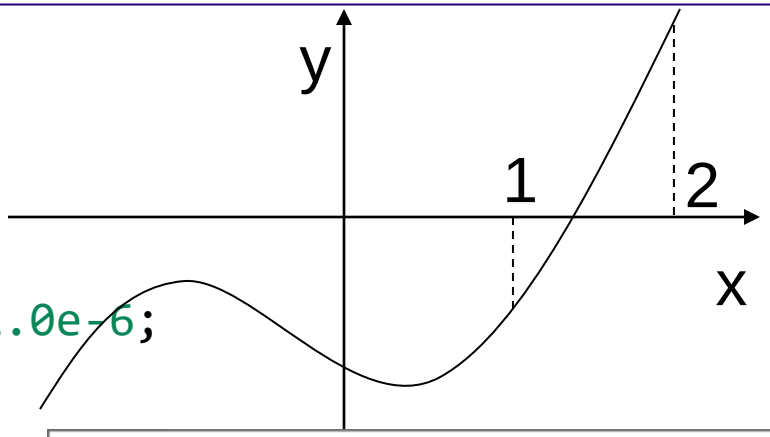
```c
#include <stdio.h>
int uscln(int a, int b) {
    while (a != b) {
        if (a > b) a = a - b;
        else b = b - a;
    }
    return a;
}
int main() {
    int A[100], N, i, r;
    printf("So phan tu : "); scanf("%d", &N);
    for (i = 0; i < N; i++) {
        printf("A[%d] = ", i + 1); scanf("%d", &A[i]);
    }
    r = A[0];
    for (i = 1; i < N; i++) r = uscln(r, A[i]);
    printf("Ket qua %d \n", r);
    return 0;
}
```

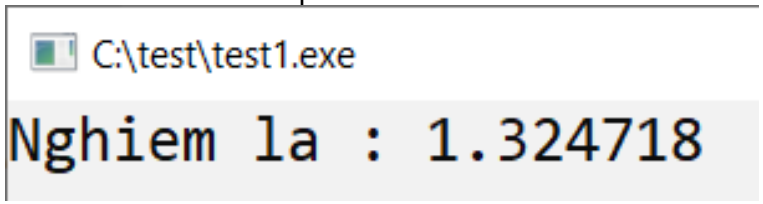
```
C:\test\test1.exe
So phan tu : 6
A[1] = 120
A[2] = 48
A[3] = 96
A[4] = 72
A[5] = 36
A[6] = 180
Ket qua 12
```

VD5: Giải phương trình f(x)=0 trên đoạn [a,b]

```c
#include <stdio.h>
#include <math.h>
float f(float x) {
    return x * x * x - x - 1;
}
int main() {
    float a = 1.0, b = 2.0, c, eps = 1.0e-6;
    do {
        c = (a + b) / 2;
        if (f(a) * f(c) < 0) b = c;
        else a = c;
    } while (fabs(b - a) > eps);
    printf("Nghiem la : %.6f", (b + a) / 2);
    return 0;
}
```

Giải phương trình $x^3-x-1=0$





VD6: Đọc tọa độ 3 điểm A,B,C và đưa ra d/tích ΔABC…

```
#include <stdio.h>
#include <math.h>
typedef struct
{
    float x, y;
} Point;

float kc(Point A, Point B)
{
    return sqrt(pow(A.x - B.x, 2) + pow(A.y - B.y, 2));
}
```

VD6: Đọc tọa độ 3 điểm A,B,C và đưa ra d/tích ΔABC…

```
int main() {
  Point A, B, C;
  float AB, BC, CA, p, S;
  printf("Toa do A (x,y): "); scanf("%f%f", &A.x, &A.y);
  printf("Toa do B (x,y): "); scanf("%f%f", &B.x, &B.y);
  printf("Toa do C (x,y): "); scanf("%f%f", &C.x, &C.y);
  AB = kc(A,B);    BC = kc(B,C);    CA = kc(C,A);
  p = (AB + BC + CA) / 2;
  S = sqrt(p * (p-AB) * (p-BC) * (p-CA));
  printf("Dien tich ABC %f", S);
  return 0;
}
```

Chương 7: Hàm

7.1. Khái niệm hàm
- Khái niệm chương trình con
- Chương trình con trong ngôn ngữ C

7.2. Khai báo và sử dụng hàm
- Khai báo và sử dụng

7.3. Phạm vi của biến
- Toàn cục và địa phương
- Biến static, biến register

7.4. Truyền tham số
- Truyền theo giá trị, truyền theo địa chỉ

Phạm vi

- Phạm vi:
  - Khối lệnh, chương trình con, chương trình chính
- Biến chỉ có tác dụng trong phạm vi được khai báo
- Trong cùng một phạm vi các biến phải có tên khác nhau.

Tình huống

- Trong hai phạm vi khác nhau có hai biến cùng tên. Trong đó một phạm vi này nằm trong phạm vi kia?

```
#include<stdio.h>
int i;
int binhphuong(int x){
  int y;
  y =  x * x;
  return y;
}
int main(){
  int y;
  for (i = 0; i <= 10; i++){
    y = binhphuong(i);
    printf("%d  ", y);
  }
  return 0;
}
```

Phân loại biến

- Biến toàn cục:
  - Biến được khai báo trong chương trình chính, được đặt sau khai báo tệp tiêu đề.
- Biến cục bộ:
  - Biến được khai báo trong lệnh khối hoặc chương trình con, được đặt trước các câu lệnh.

Ghi chú:

  - Hàm **main()** cũng là một chương trình con nhưng là nơi chương trình được bắt đầu.
  - Biến khai báo trong hàm **main()** cũng là biến cục bộ, chỉ có phạm vi trong hàm **main()**.

Biến static

- Biến cục bộ ra khỏi phạm vi thì bộ nhớ dành cho biến được giải phóng.
- Yêu cầu lưu trữ giá trị của biến cục bộ một cách lâu dài => sử dụng từ khóa ***static***

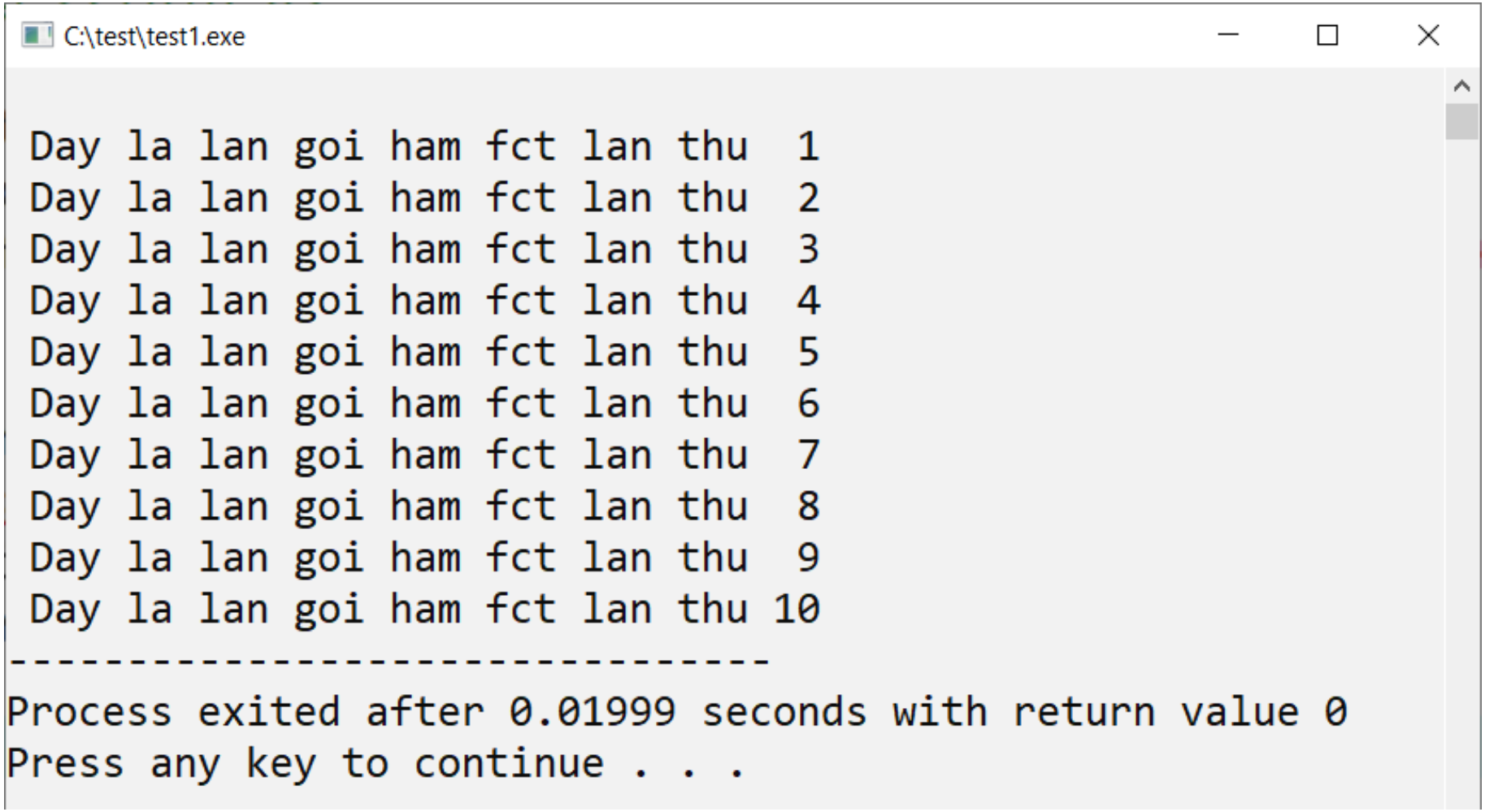
Cú pháp:

static <kiểu_dữ_liệu> tên_biến;

Ví dụ

```c
#include <stdio.h>

void fct() {
  static int count = 1;
  printf("\n Day la lan goi ham fct lan thu %2d", count++);
}

int main() {
  int i;
  for (i = 0; i < 10; i++)  fct();
  return 0;
}
```

Ví dụ → Kết quả


C:\test\test1.exe
Day la lan goi ham fct lan thu 1
Day la lan goi ham fct lan thu 2
Day la lan goi ham fct lan thu 3
Day la lan goi ham fct lan thu 4
Day la lan goi ham fct lan thu 5
Day la lan goi ham fct lan thu 6
Day la lan goi ham fct lan thu 7
Day la lan goi ham fct lan thu 8
Day la lan goi ham fct lan thu 9
Day la lan goi ham fct lan thu 10

Process exited after 0.01999 seconds with return value 0
Press any key to continue . . .

Biến register

- Thanh ghi có tốc độ truy cập nhanh hơn RAM, bộ nhớ ngoài
- Lưu biến trong thanh ghi sẽ tăng tốc độ thực hiện chương trình

Cú pháp:

```
register <kiểu_dữ_liệu> tên_biến;
```

Lưu ý: Số lượng biến register không nhiều và thường chỉ với kiểu dữ liệu nhỏ như int, char

Chương 7: Hàm

7.1. Khái niệm hàm
- Khái niệm chương trình con
- Chương trình con trong ngôn ngữ C

7.2. Khai báo và sử dụng hàm
- Khai báo và sử dụng

7.3. Phạm vi của biến
- Toàn cục và địa phương
- Biến static, biến register

7.4. Truyền tham số
- Truyền theo giá trị, truyền theo địa chỉ

VD1: Viết hàm đổi giá trị 2 biến

```c
#include <stdio.h>
void swap(int a, int b) {
  int x = a;
  a = b;
  b = x;
}
int main() {
  int a = 5, b = 100;
  printf("Truoc: a=%d, b=%d \n\n", a, b);
  swap(a, b);
  printf("Sau  : a=%d, b=%d \n\n", a, b);
  return 0;
}
```

Truyền theo giá trị và truyền theo biến

- Truyền theo trị
  - Dựa trên nguyên tắc truyền những bản sao của biến được truyền
  - Những câu lệnh thay đổi giá trị tham số hình thức sẽ không ảnh hưởng tới biến được truyền
- Truyền theo biến
  - Tham số được truyền sẽ thực sự là biến và các thao tác sẽ thi hành trực tiếp với biến
  - Những câu lệnh thay đổi giá trị tham số hình thức sẽ ảnh hưởng tới biến được truyền

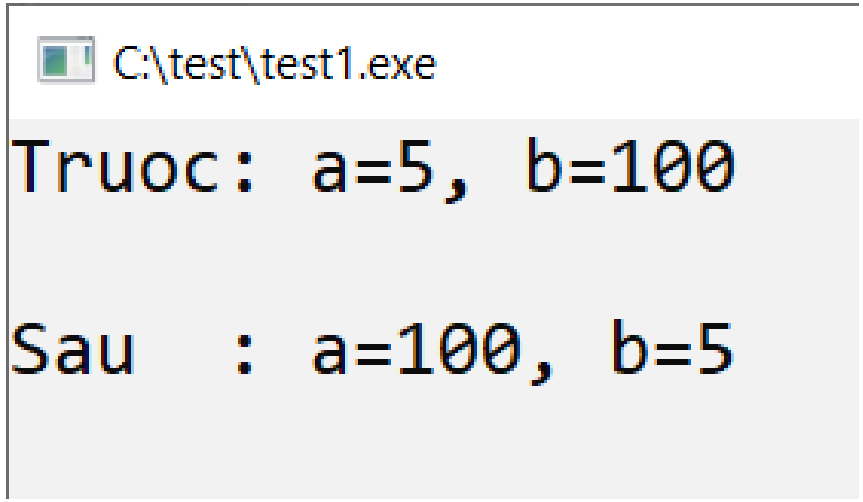
Truyền theo biến

- Thực chất là truyền theo địa chỉ của biến
- Khi khai báo hàm [tham số có kiểu "*địa chỉ*"]:
  - Khai báo là một con trỏ, trỏ tới một đối tượng có kiểu muốn truyền vào.
  - Ví dụ: void swap (int *pa, int *pb);
- Khi truyền tham số
  - Địa chỉ của biến được truyền

    Ví dụ: swap(&a, &b)
  - Có thể truyền tên của mảng
    - Tên mảng là hằng địa chỉ

VD2: Truyền theo địa chỉ của biến

```c
#include <stdio.h>

void swap(int * pa, int * pb) {
  int x = *pa;
  *pa = *pb;
  *pb = x;
}

int main() {
  int a = 5, b = 100;
  printf("Truoc: a=%d, b=%d \n\n", a,b);
  swap(&a, &b);
  printf("Sau  : a=%d, b=%d \n\n", a, b);
  return 0;
}
```



VD3: Viết hàm trả về giá trị lớn nhất và nhỏ nhất của mảng

Câu hỏi 1: Kết quả đưa ra màn hình

```
#include<stdio.h>
void fun(int n) {
  if(n > 0) {
    fun(--n);
    printf("%d ", n);
    fun(--n);
  }
}
int main() {
  fun(3);
  return 0;
}
```

a	0 2 1 0
b	0 1 0 2
c	1 1 2 0
d	0 1 2 0
e	0 2 0 1

Câu hỏi 2: Kết quả đưa ra màn hình

```
#include<stdio.h>
void fun(int *i, int *j) {
   *i = *i * *i;
   *j = *j * *j;
}
int main() {
   int i=5, j=2;
   fun(&i, &j);
   printf("%d, %d", i, j);
   return 0;
}
```

a	5, 2
b	2, 5
c	10, 4
d	4, 25
e	25, 4

Câu hỏi 3: Kết quả đưa ra màn hình

```
#include<stdio.h>
void fun(char*);
int main() {
   char a[10]="ABCDEF";
   fun(&a[0]);
   return 0;
}
void fun(char *a){
   printf("%c", *++a);
   a++;
   printf("%c", *a);
}
```

a	AB
b	AC
c	BC
d	BD
e	CD

Hàm đệ quy

- Là hàm mà nội dung có lời gọi đến chính nó
- Thường sử dụng cho các thuật toán đệ quy
  - Trường hợp cơ bản: kết thúc hàm, không cần gọi lại
  - Trường hợp tổng quát: có lời gọi lại hàm với tham số ở cấp độ nhỏ hơn

Hàm đệ quy

- **Ví dụ 1:** Tính giai thừa n!

```
#include <stdio.h>
int giaithua(int k) {
    return k == 1 ? 1 : k * giaithua(k - 1);
}
int main() {
    int n;
    printf("Nhap n: ");
    scanf("%d", &n);
    printf("%d! = %d", n, giaithua(n));
    return 0;
}
```

Hàm đệ quy

- **Ví dụ 2:** In ra dãy số Fibonaci

```c
#include <stdio.h>
int fibo(int k) {
    if (k == 0) return 0;
    if (k == 1) return 1;
    return fibo(k - 1) + fibo(k - 2);
}
int main() {
    int i, n;
    printf("Nhap n: "); scanf("%d", &n);
    printf("So fibonaci thu %d la: %d\n", n, fibo(n));
    printf("Day so fibonaci\n");
    for (i = 0; i <= n; i++)
        printf("%5d", fibo(i));
    return 0;
}
```

Hàm đệ quy

- **Ví dụ 3:** In ra biểu diễn nhị phân của số nguyên dương N

```
#include <stdio.h>
void bin(int k) {
    if (k > 0) {
        bin(k / 2);
        printf("%d", k % 2);
    }
}
void main() {
    int n;
    printf("Nhap n: ");
    scanf("%d", &n);
    printf("Bieu dien nhi phan: ");
    bin(n);
}
```

Bài tập

1. Viết các hàm tính diện tích và chu vi hình tròn.
2. Viết hàm kiểm tra 1 số có phải là số nguyên tố hay không.
3. Viết hàm nhập/xuất mảng a các số nguyên có n phần tử từ bàn phím.
4. Viết hàm tìm số lớn nhất trong mảng a
5. Viết hàm thực hiện sắp xếp mảng a theo thứ tự tăng dần hoặc giảm dần.

Bài tập

6. Viết hàm kiểm tra 2 xâu ký tự có giống nhau không (không phân biệt chữ hoa chữ thường).
7. Viết hàm tính khoảng cách giữa 2 điểm trong không gian 3 chiều.
8. Viết hàm tìm trung điểm của đoạn thẳng tạo bởi 2 điểm trong không gian 3 chiều.
9. Viết chương trình tính X^n sử dụng thuật toán đệ quy.
10. Viết chương trình tìm USCLN của 2 số nguyên dương sử dụng thuật toán đệ quy.

Bài 11

Cho hàm f(x) được định nghĩa như sau

$$f(x) = \begin{cases} \sqrt{e^{2x+1} + 1} + 7 & khi\ |x| \le 2 \\ x^5 + 5x^3 + x + 1 & khi\ |x| > 2 \end{cases}$$

Hãy viết chương trình thực hiện các công việc sau

- Viết chương trình con tính hàm trên
- Nhập vào từ bàn phím 2 số thực x, y, tính và đưa ra màn hình $(f(x)+f(y))^2$
- Đưa ra màn hình theo dòng các cặp <*x, f(x)*> *(định dạng đưa ra là các số thực tĩnh có 2 chữ số sau dấu chấm*) trong đó giá trị của x lần lượt là -5.0; -4.9; -4.8; … 2.8; 2.9; 3.0.

Bài 12

- Tiền điện được tính theo số điện tiêu thụ như sau
  - Dùng ít hơn 250 số: 2000đồng/số
  - Dùng từ 250 đến 400 số: 3000 đồng/số
  - Dùng từ 400 đến 500 số: 4000 đồng/số
  - Dùng từ 500 số trở lên: 5000đ/số
- Hãy viết hàm *TienDien*, trả về số tiền điện phải trả với tham số vào là số điện năng đã tiêu thụ
- Nhập vào danh sách sử dụng điện của các hộ gia đình (*tên chủ hộ, số điện tiêu thụ*), Kết thúc nhập khi đã đủ 100 hộ hoặc nhập tên chủ hộ là "***"
- Đưa ra màn hình hộ trả tiền điện ít nhất
- Sắp xếp danh sách theo thứ tự tăng của tiền phải trả

Bài 13

Cho hàm f(x) được định nghĩa như sau

$$f(x) = \begin{cases} \sqrt[3]{4 - x^2} + 1 & Khi\ |x| < 2 \\ 7 & Khi\ |x| = 2 \\ e^{x^3+1} + \log_5(x^2 - 1) & Khi\ |x| > 2 \end{cases}$$

Hãy viết chương trình thực hiện các công việc sau

- Viết chương trình con tính hàm trên
- Nhập từ bàn phím N số thực và tìm và đưa ra màn hình giá trị lớn nhất của hàm f(x) tại N số đã nhập.
- Nhập vào từ bàn phím một dãy các số thực cho tới khi nhập đủ 100 số hoặc cho tới khi gặp một giá trị x thỏa mãn f(x) lớn hơn giá trị lớn nhất trong câu trên.