



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

TIN HỌC ĐẠI CƯƠNG

Phần 2. Lập trình

Bài 7. Hàm

Nội dung

7.1. Khái niệm hàm

7.2. Khai báo và sử dụng hàm

7.3. Phạm vi của biến

7.4. Hàm đệ quy

Nội dung

7.1. Khái niệm hàm

7.1.1. Khái niệm chương trình con

7.1.2. Phân loại chương trình con

7.2. Khai báo và sử dụng hàm

7.3. Phạm vi của biến

7.4. Hàm đệ quy

7.1.1. Khái niệm chương trình con

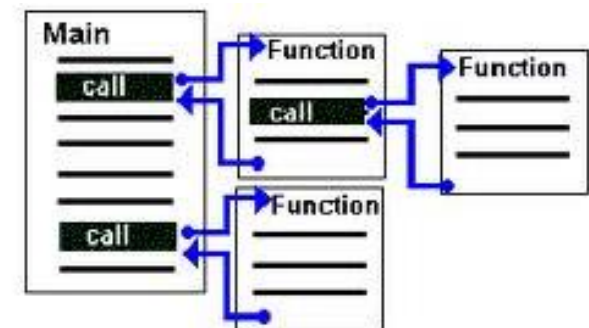
- Khái niệm
 - Là một chương trình nằm trong một chương trình lớn hơn nhằm thực hiện một nhiệm vụ cụ thể
- Vai trò
 - Chia nhỏ chương trình ra thành từng phần để quản lý => **Phương pháp lập trình có cấu trúc**
 - Có thể sử dụng lại nhiều lần: printf, scanf...
 - Chương trình dễ dàng đọc và bảo trì hơn



KT tự cung tự cấp

><

KT hàng hoá

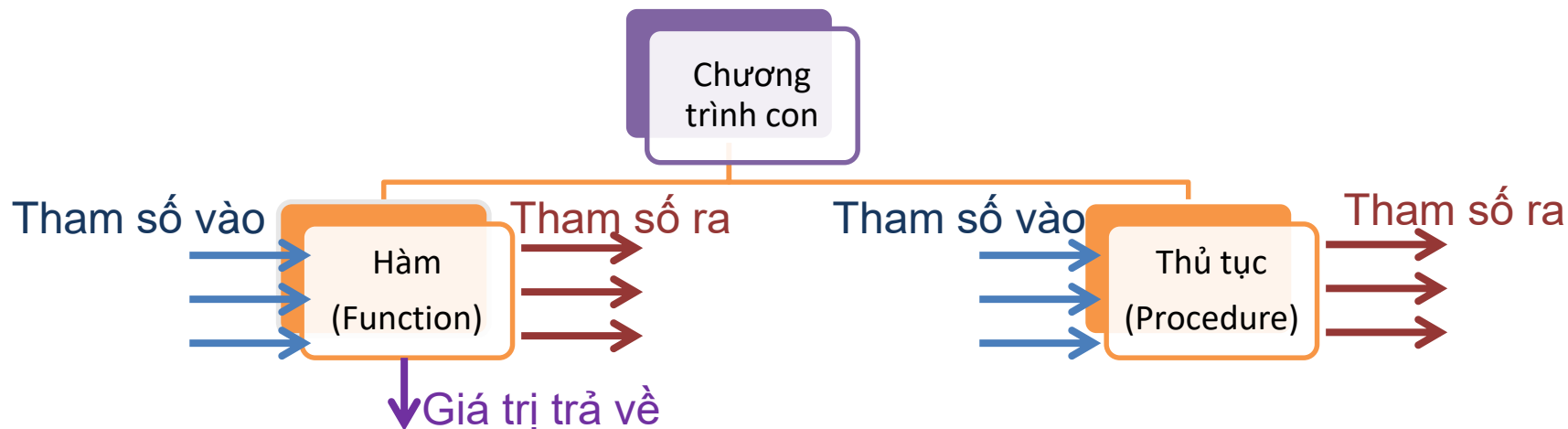


Ví dụ

```
#include <stdio.h>
#include <conio.h>
void main() {
    printf("Hello World\n");
    getch();
}
```

7.1.2. Phân loại chương trình con

- Phân loại chương trình con

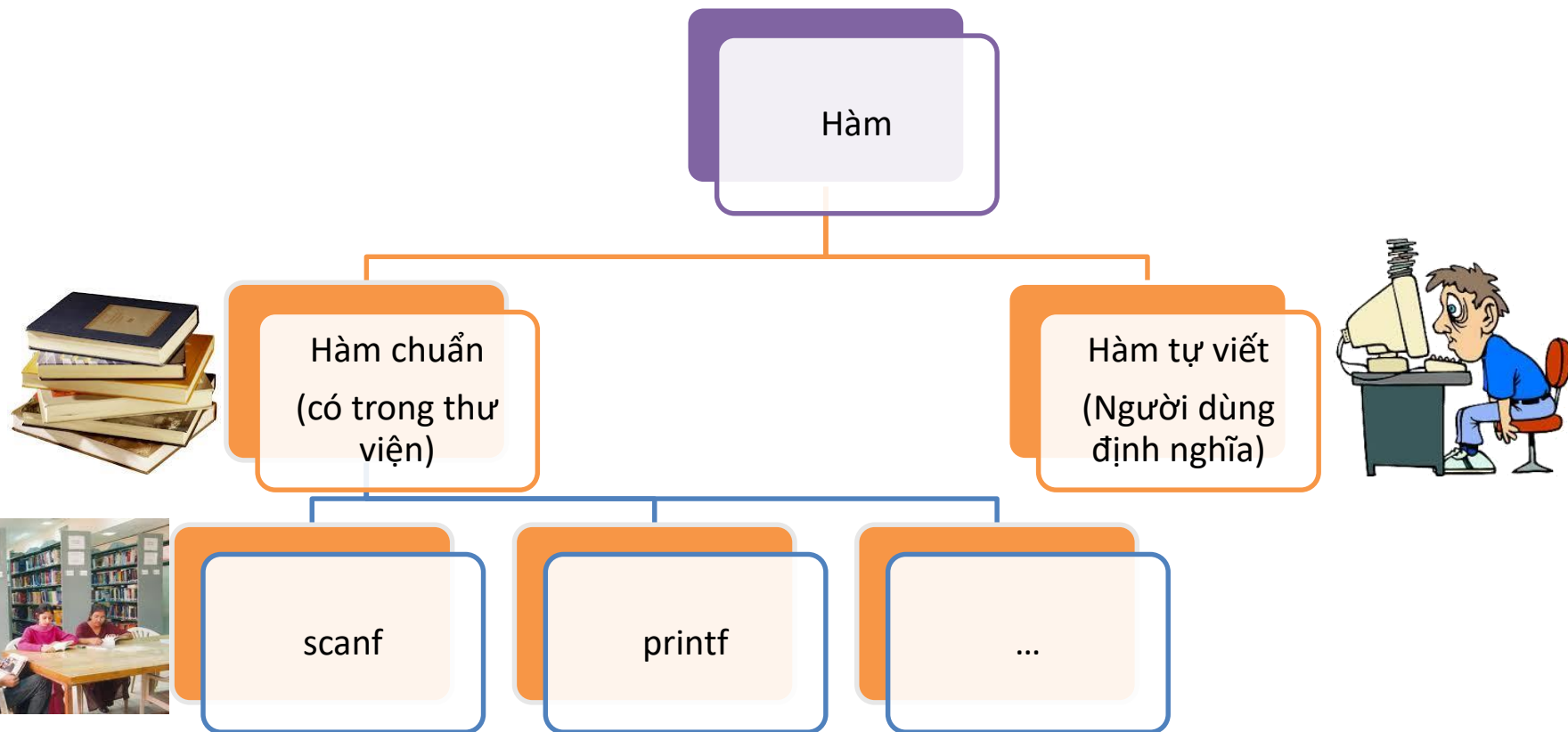


– Hàm: **trả về giá trị trong khi thủ tục thì không**

– Trong C:

- Chỉ cho phép khai báo chương trình con là hàm.
- Sử dụng kiểu “**void**” với ý nghĩa “**không là kiểu dữ liệu nào cả**” để chuyển thủ tục về dạng hàm

- Phân loại hàm



7.2. Khai báo và sử dụng hàm

7.2.1. Khai báo hàm

7.2.2. Sử dụng hàm

7.2.1. Khai báo hàm

- Ví dụ:
 - Chương trình in ra bình phương của các số tự nhiên từ 1 đến 10
 - Gồm 2 hàm:
 - Hàm *binhphuong*(*int x*): trả về bình phương của x
 - Hàm *main*(): với mỗi số nguyên từ 1 đến 10, gọi hàm *binhphuong* với một giá trị đầu vào và hiển thị kết quả.


7.2.1. Khai báo hàm

Khai báo hàm

Gọi hàm

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

7.2.1. Khai báo hàm



```
[<kiểu_giá_trị_trả_về>] tên_hàm ([danh_sách_tham_số])  
{  
    [<Các_khai_báo>]  
    [<Các_câu_lệnh>]  
}
```

- Dòng đầu hàm

- Là thông tin trao đổi giữa các hàm. Phân biệt giữa các hàm với nhau. Bao gồm:
 - Kiểu giá trị trả về: kiểu dữ liệu bất kì, con trỏ, nhưng không được là kiểu dữ liệu mảng.
 - Tên hàm: là tên hợp lệ, trong C tên hàm là duy nhất

7.2.1. Khai báo hàm

– Tham số

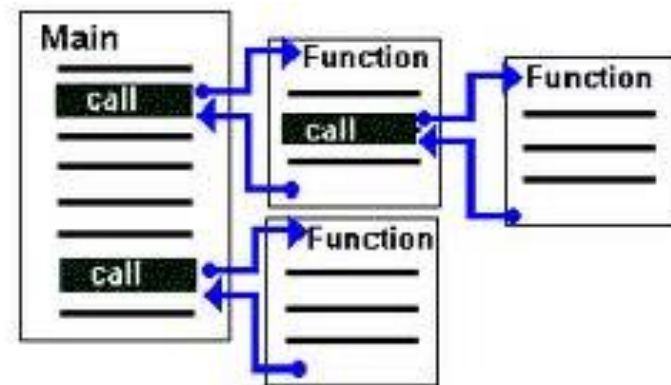
- Cho biết những tham số giả định cung cấp hoạt động cho hàm => các tham số hình thức
- Tham số cung cấp dữ liệu cho hàm lúc hoạt động => tham số thực

– Ví dụ: `int max(int a, int b, int c)`

• Thân hàm

• Lệnh **return**

- Sau khi thực hiện xong, trở về điểm mà hàm được gọi thông qua câu lệnh **return** hoặc kết thúc hàm.
- Cú pháp chung: ***return biểu_thức;***



Ví dụ khai báo hàm

- Ví dụ: hàm tính giai thừa

```
int giai_thua(int a)
{
```

————> Dòng đầu hàm

```
    int ket_qua;
    int i;
```

————> Các khai báo

```
    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;
}
```

————> 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)
```

————> Dòng đầu hàm

```
{
```

```
    int ket_qua;
```

————> Các khai báo

```
    ket_qua = a;
```

```
    if(ket_qua < b) ket_qua = b;
```

```
    if(ket_qua < c) ket_qua = c;
```

```
    return ket_qua;
```

————> Các câu lệnh

```
}
```

Ví dụ khai báo hàm

- Ví dụ: hàm (thủ tục) vẽ tam giác *

```
void tamgiac(int n)
```

————> Dòng đầu hàm

```
{
```

```
    int i, j, k;
```

————> Các khai báo

```
    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");
```

```
    }
```

————> Các câu lệnh

```
}
```

Ví dụ khai báo hàm

- Ví dụ: hàm (thủ tục) vẽ tam giác *

```
void tamgiac()
```

————> Dòng đầu hàm

```
{
```

```
    int i, j, k, n = 10;
```

————> Các khai báo

```
    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");
```

```
    }
```

————> Các câu lệnh

```
}
```

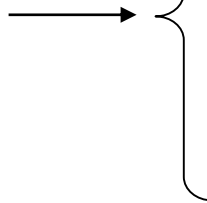

7.2.1. Khai báo hàm

Nguyên mẫu hàm
(function
prototype)



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

Định nghĩa hàm



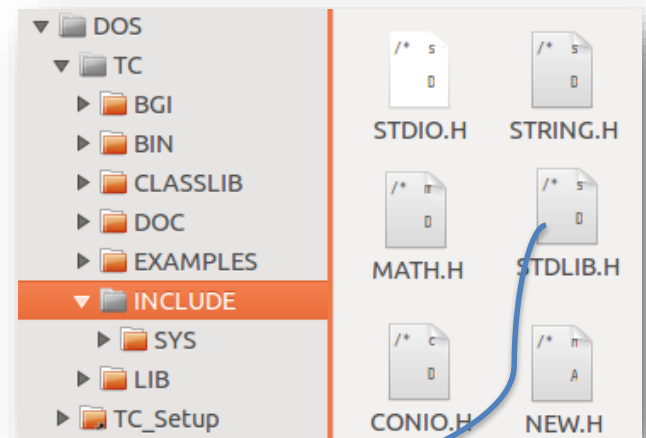
```
int binhphuong(int x) {
    int y;
    y = x * x;
    return y;
}
```

7.2.1. Khai báo hàm

- Ý nghĩa của nguyên mẫu hàm
 - Cho phép định nghĩa sau khi sử dụng. Nhưng phải khai báo trước.
 - Cho phép đưa ra lời gọi đến một hàm mà không cần biết định nghĩa.
 - Ví dụ: khi gọi **printf**, **scanf** chúng ta chỉ cần quan tâm các tham số truyền cho hàm
 - Tập **stdio.h** chứa nguyên mẫu hàm của **printf** và **scanf**

7.2.1. Khai báo hàm

- 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_tieu_de>**
 - Tập_tieu_de (.h) đã chứa các nguyên mẫu hàm



```
154 size_t _cdecl fwrite(const void *__ptr, size_t __size, size_t __n,
155 FILE *__stream);
156 char *_cdecl gets(char *__s);
157 void _cdecl perror(const char *__s);
158 int _cdecl printf(const char *__format, ...);
159 int _cdecl puts(const char *__s);
160 int _ctype remove(const char *__path);
161 int _ctype rename(const char *__oldname, const char *__newname);
162 void _cdecl rewind(FILE *__stream);
163 int _cdecl scanf(const char *__format, ...);
164 void _cdecl setbuf(FILE *__stream, char *__buf);
165 int _cdecl setvbuf(FILE *__stream, char *__buf,
166 int __type, size_t __size);
167 int _cdecl sprintf(char *__buffer, const char *__format, ...);
168 int _cdecl sscanf(const char *__buffer,
169 const char *__format, ...);
170 char *_cdecl strerror(int __errnum);
171 FILE *_cdecl tmpfile(void);
```

rewind(FILE *__stream);
scanf(const char *__format, ...);
setvbuf(FILE *__stream, char *__buf, int __type, size_t __size);

perror(const char *__s);
printf(const char *__format, ...);
puts(const char *__s);

7.2.2. Sử dụng hàm

- Cú pháp:

tên_hàm(**danh_sách_tham_số**);

- Ví dụ: binhphuong(0),
binhphuong(1)...

- Lưu ý:


- Nếu hàm nhận nhiều tham số thì các tham số ngăn cách nhau bởi dấu phẩy
- Luôn luôn cần cặp dấu ngoặc () sau tên hàm
- Các tham số của hàm sẽ nhận các giá trị từ tham số truyền vào
- Thực hiện lần lượt các lệnh cho đến khi gặp lệnh return/kết thúc chương trình

```
printf("%d %s",4,"abc");
```

```
scanf("%d %s", &x, s);
```

```
xyz();
```

abc(0, 4)



```
abc(int x, float y)  
{  
    return (...);  
}
```

Ví dụ khai báo hàm

- Ví dụ: hàm tính giai thừa

```
int giai_thua(int a)
{
```

————> Dòng đầu hàm

```
    int ket_qua;
    int i;
```

————> Các khai báo

```
    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;
}
```

————> Các câu lệnh

Ví dụ

- Cách sử dụng hàm

```
g = giaiithua(10) ;
```

```
g = giaiithua(n) ;
```

```
m = max(10, 20, 30) ;
```

```
m = max(a, b, c) ;
```

```
tamgiac(10) ;
```

```
tamgiac(n) ;
```

```
tamgiac() ;
```

Ví dụ

- Chương trình tính số tổ 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>
#include <conio.h>
#include <stdlib.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;
}
```

Ví dụ

- Viết hàm đếm số ký tự chữ hoa, chữ thường và chữ số có trong xâu ký tự.

7.3. Phạm vi của biến

7.3.1. Phạm vi của biến

7.3.2. Phân loại biến

7.3.3. Câu lệnh static và register

7.3.1. Phạm vi của biến

- Phạm vi:
 - khối lệnh,
 - chương trình con,
 - chương trình chính
- Biến khai báo trong phạm vi nào thì sử dụng trong phạm vi đó
- Trong cùng một phạm vi các biến 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>
#include<conio.h>
int i;

int binhphuong(int x)
{
    int y;
    y = x * x;
    return y;
}

void main()
{
    int y;
    for (i=0; i<= 10; i++)
    {
        int k;
        y = binhphuong(i);
        printf("%d  ", y);
    }
}
```

Chương
trình
con

Ch.trình
chính

Khối
Lệnh

7.3.2. Phân loại biến

- Phân loại biến
 - **Biến toàn cục:** biến được khai báo ngoài mọi hàm, được sử dụng ở các hàm đứng sau nó
 - **Biến cục bộ:** biến được khai báo trong lệnh khối hoặc chương trình con, được đặt trước các câu lệnh.
- Chú ý
 - Hàm **main()** cũng là một chương trình con nhưng là nơi chương trình được bắt đầu cũng như kết thúc
 - Biến khai báo trong hàm **main()** cũng là biến cục bộ, chỉ có phạm vi trong hàm **main()**.

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

Ví dụ phạm vi biến

- Ví dụ 1:

```
#include <stdio.h>
```

```
void main()
```

```
{1
```

```
{2 int a = 1;
```

```
printf("\n a = %d", a);
```

```
{3 int a = 2;
```

```
printf("\n a = %d", a);
```

```
}3
```

```
printf("\n a = %d", a);
```

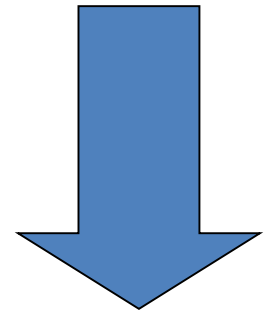
```
}2
```

```
{2' int a = 3;
```

```
printf("\n a = %d", a);
```

```
}2'
```

```
}1
```



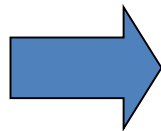
```
a1 = 1
a2 = 2
a3 = 1
a4 = 3_
```

Ví dụ 2

```
#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;
}
```

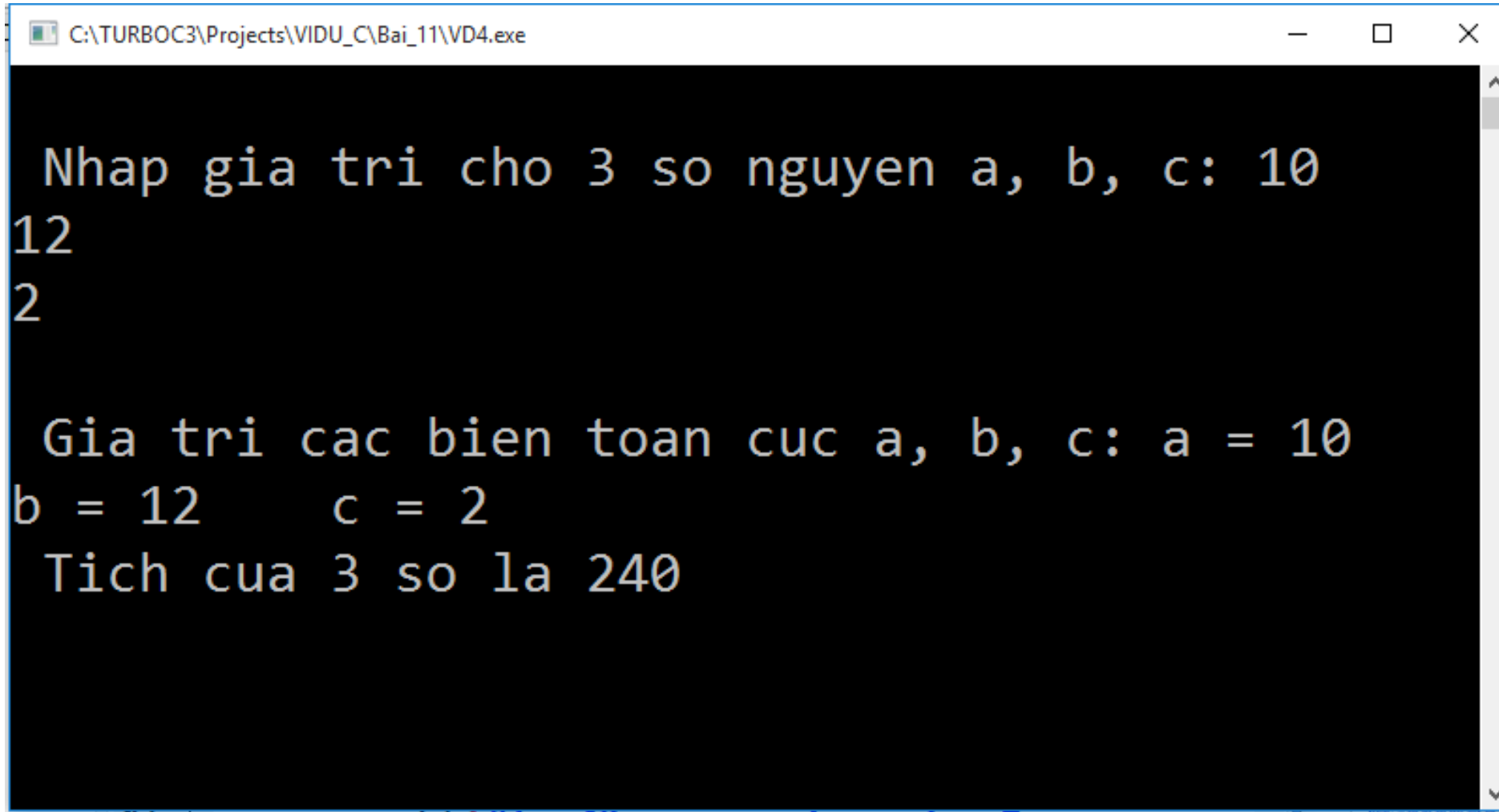

Ví dụ 2

```
int 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());  
    return 0;  
}
```



```
Nhap 3 so nguyen a, b, c: 2 3 4  
Gia tri cac bien tong the a, b, c:  
a = 2      b = 3      c = 4  
Tich cua 3 so la 24
```

Ví dụ 2



The screenshot shows a Turbo Pascal IDE window titled "C:\TURBOC3\Projects\VIDU_C\Bai_11\VD4.exe". The window contains a text area with the following text:

```
Nhap gia tri cho 3 so nguyen a, b, c: 10
12
2

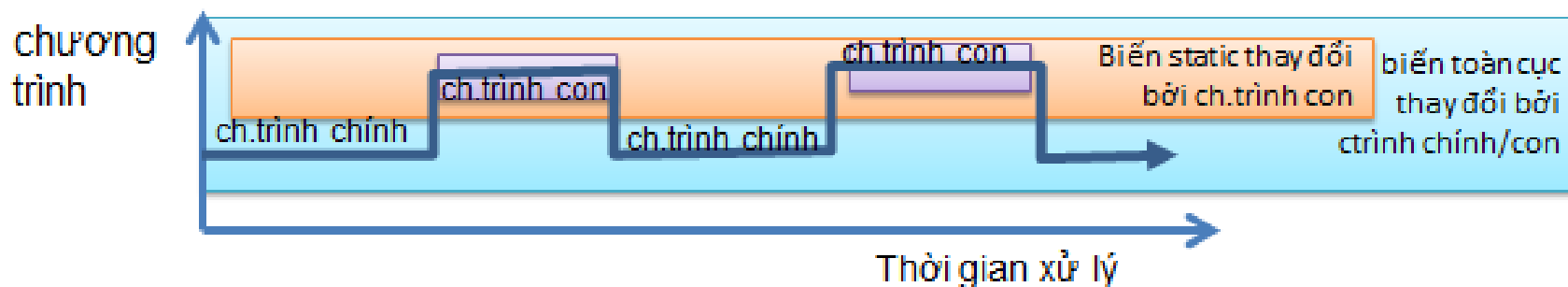
Gia tri cac bien toan cuc a, b, c: a = 10
b = 12      c = 2
Tich cua 3 so la 240
```

The text is displayed in a monospaced font, with the first line being a prompt and the subsequent lines being user input and program output. The output shows the values of variables a, b, and c, and the result of their product.

7.3.3. Câu lệnh static và register

- **Biến static**

- Xuất phát: 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**
- So sánh với biến toàn cục?



- **Cú pháp:**

static <kiểu_dữ_liệu> tên_biến;

7.3.3. Câu lệnh static và register

```
#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();
}
```

7.3.3. Câu lệnh static và register

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

7.3.3. Câu lệnh static, register

- 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

7.4. 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

7.4. Hàm đệ quy

- **Ví dụ 1:** Tính giai thừa $n!$

```
#include <stdio.h>

int giai thua(int k) {
    return k == 1 ? 1 : k * giai thua(k - 1);
}

int main() {
    int n;
    printf("Nhap n: ");
    scanf("%d", &n);
    printf("%d! = %d", n, giai thua(n));
    return 0;
}
```


7.4. Hàm đệ quy

- **Ví dụ 2:** In ra dãy số Fibonacci

```
#include <stdio.h>
```

```
int fibo(int k)
```

```
{
```

```
    if (k == 0)
```

```
        return 0;
```

```
    if (k == 1)
```

```
        return 1;
```

```
    return fibo(k - 1) + fibo(k - 2);
```

```
}
```

7.4. Hàm đệ quy

- **Ví dụ 2: In ra dãy số Fibonacci**

```
int main()
{
    int i, n;
    printf("Nhap n: "); scanf("%d", &n);

    printf("So fibonacci thu %d la: %d\n", n,
fibonacci(n));

    printf("Day so fibonacci\n");
    for (i = 0; i <= n; i++)
        printf("%5d", fibonacci(i));

    return 0;
}
```

7.4. 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.