

ÔN TẬP

NGÔN NGỮ LẬP TRÌNH C CƠ BẢN



GIỚI THIỆU NGÔN NGỮ C VÀ BORLAND C

Giảng Viên: Th.S Dương Thành Phết

Email: phetcm@gmail.com

Website: <http://www.thayphet.net>

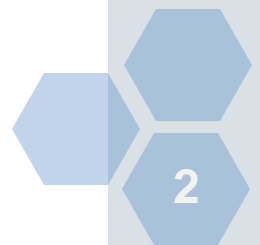
Tel: 0918158670





NỘI DUNG

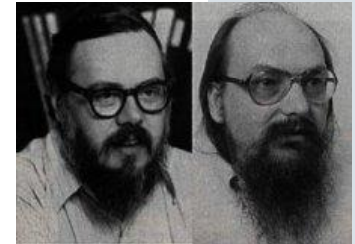
1. Tổng quan về ngôn ngữ lập trình C
2. Môi trường lập trình Borland C





1. TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH C

1.1. Lịch sử ra đời



C là ngôn ngữ lập trình rất mạnh và được sử dụng rộng rãi khắp thế giới.

Ngôn ngữ **C** (Tiền thân là ngôn ngữ B) được phát triển từ đầu **thập niên 1970** bởi **Ken Thompson** và **Dennis Ritchie** tại phòng thí nghiệm **Bell Telephone** (Mỹ) vào năm 1972.

Nhưng sau đó nhiều nhà lập trình tung ra rất nhiều phiên bản C. cuối cùng viện định chuẩn quốc gia Mỹ đã họp và định ra chuẩn cho ngôn ngữ C gọi là **AnSi C**.

C là ngôn ngữ lập trình cô đọng chỉ chứa các từ khóa với nhiều hàm được tạo sẵn, hỗ trợ nhiều phép toán.



1. TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH C

1.2. Đặc điểm ngôn ngữ C

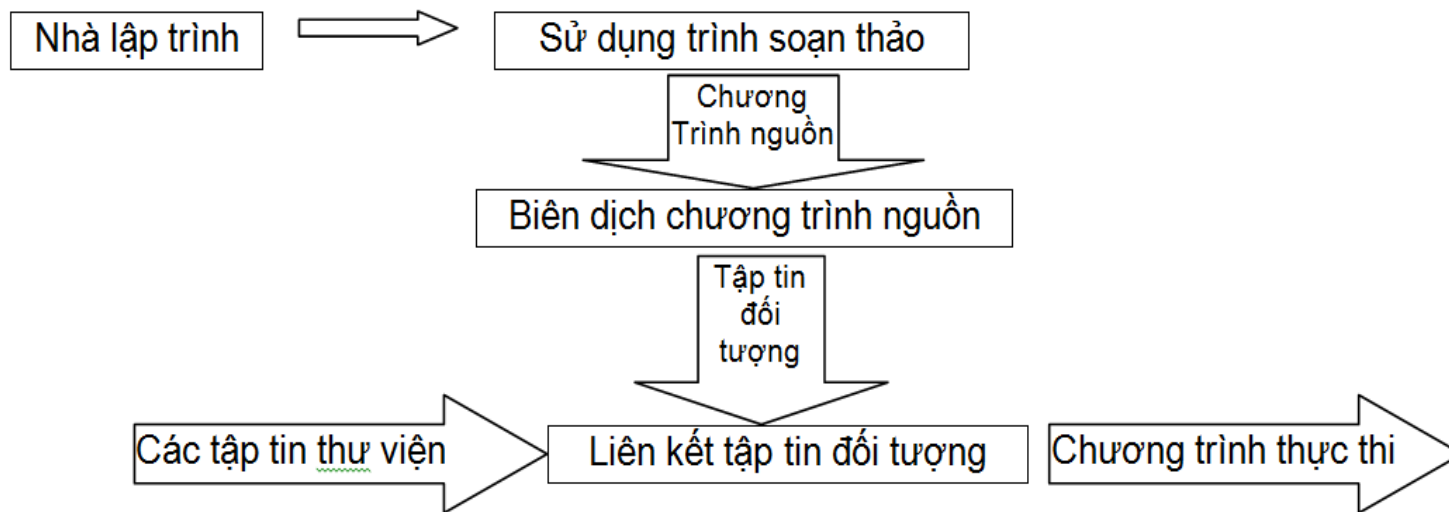
- ✓ **Tính cô đọng:** Chỉ có 32 từ khóa và 40 toán tử
- ✓ **Tính cấu trúc:** tập hợp các chỉ thị có cấu trúc lựa chọn, lặp . . Tổ chức rõ ràng, dễ hiểu
- ✓ **Tính tương thích:** Có bộ tiền xử lý và thư viện chuẩn phong phú dễ dàng chuyển sang máy khác
- ✓ **Tính linh động:** Uyển chuyển chấp nhận nhiều cú pháp thể hiện, dễ thu gọn mã lệnh để chạy nhanh hơn
- ✓ **Biên dịch:** Biên dịch nhiều tập tin chương trình thành các tập in đối tượng và liên kết lại để thực thi





1. TỔNG QUAN VỀ NGÔN NGỮ LẬP TRÌNH C

1.3. Chu trình triển khai chương trình C



Bước 1: Tạo tập tin .C hoặc .CPP

Bước 2 : Tạo tập tin .OBJ

Bước 3: Tạo tập tin .EXE

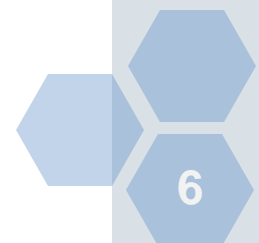
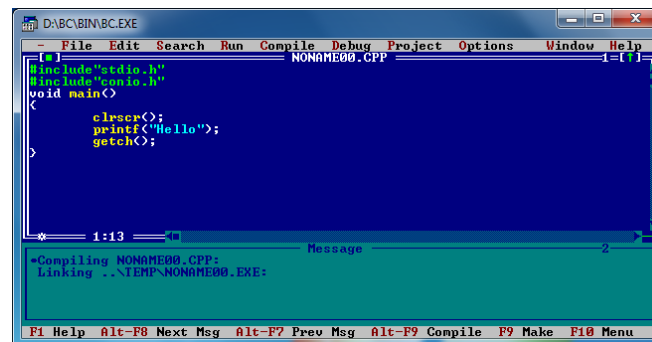
Bước 4 : Chạy chương trình



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

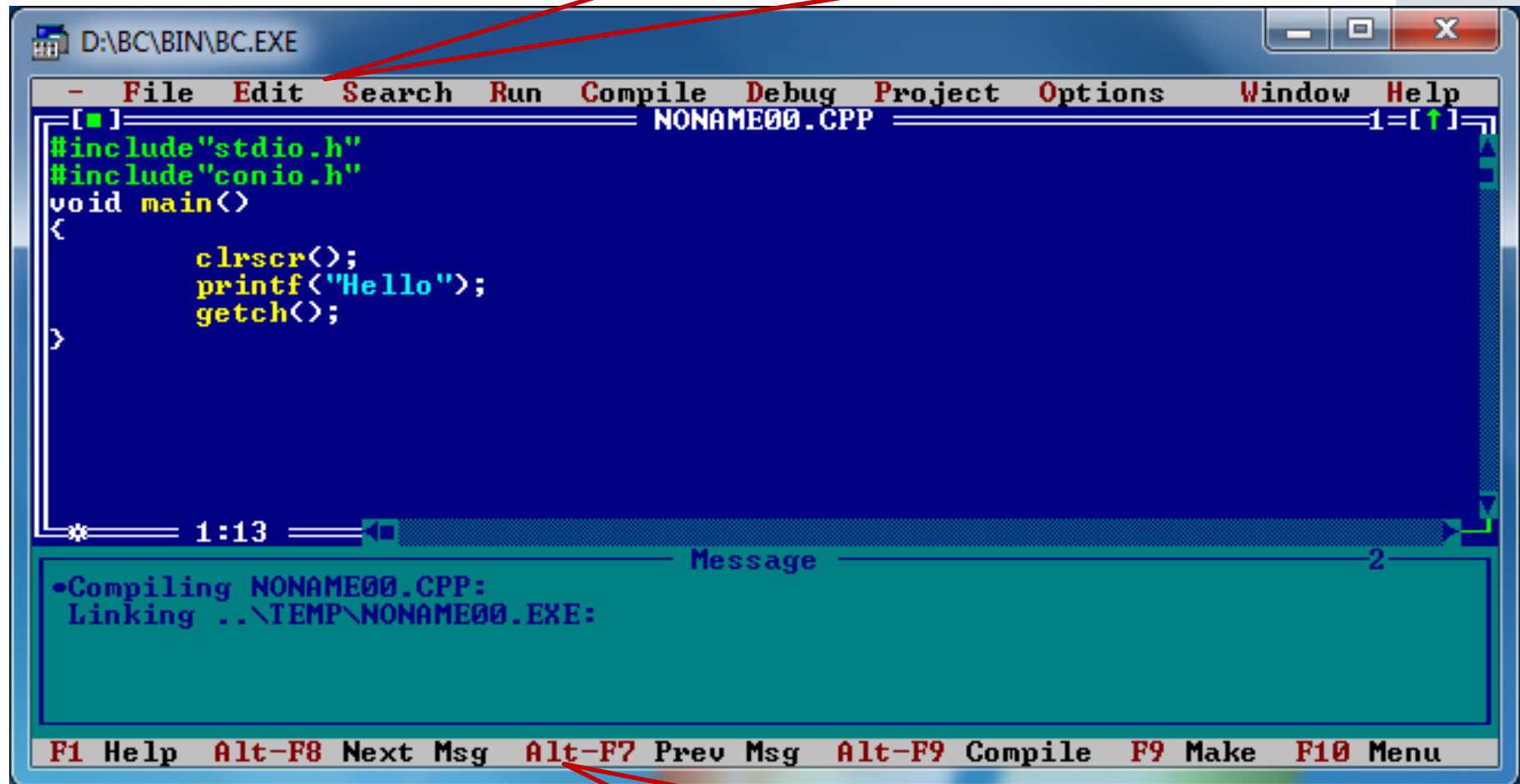
2.1. Giới thiệu Borland C

- ✓ Borland C là môi trường lập trình (trình biên dịch) do hãng borland cung cấp cho phép soạn thảo và biên dịch chương trình C.
- ✓ Có rất nhiều phiên bản for Dos hoặc for Win
- ✓ Phiên bản ứng dụng là Borland C 3.1 for Dos (tùy theo Windows có các bản khác nhau)
- ✓ Cài đặt: Download BC.exe và giải nén vào ổ đĩa D:
- ✓ Khởi động chương trình: chạy file **BC.exe** trong **/BC/Bin**



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

Menu bar(F10 hoặc phím tắt: ALT+F, ALT+R, ..)



Phím tắt 1 số chức năng đặc biệt

2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.2. Soạn thảo chương trình

✓ Tạo mới: File/New

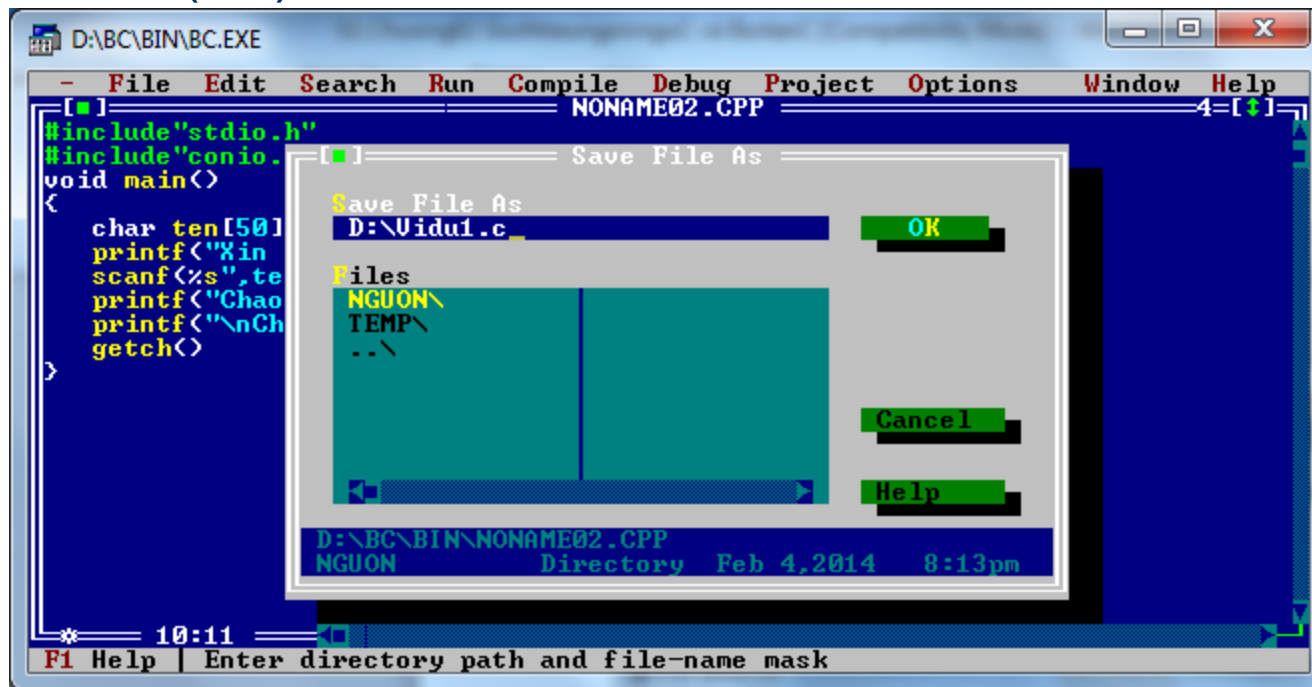
```
D:\ABC\BIN\BC.EXE
- File Edit Search Run Compile Debug Project Options Window Help
VIDU1.C
#include "stdio.h"
#include "conio.h"
void main()
{
    char ten[50];
    clrscr();
    printf("Xin cho biet ten:");
    scanf("%s",ten);
    printf("Chao ban %s", ten);
    printf("\nChao mung ban den voi ngon ngu lap trinh C");
    getch();
}
5:23
F1 Help Alt-F8 Next Msg Alt-F7 Prev Msg Alt-F9 Compile F9 Make F10 Menu
```

Soạn thảo chương trình

2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.3. Lưu chương trình

✓ File/Save (F2)



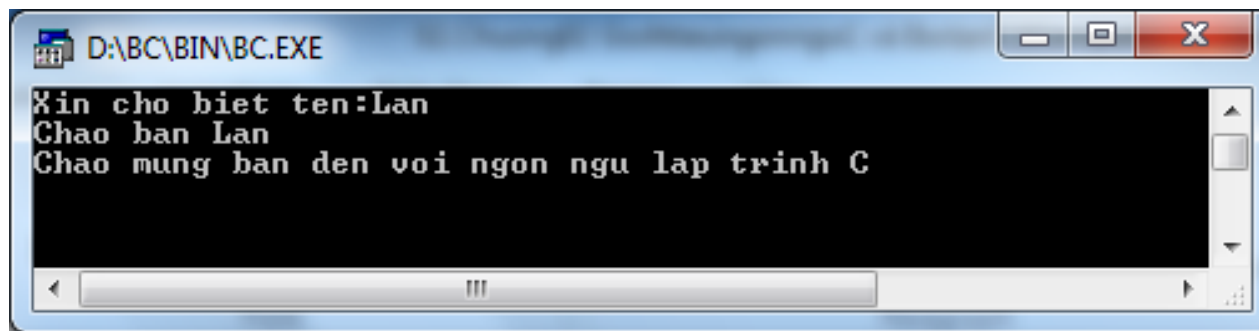
✓ Phần tên bắt đầu là ký tự, không có khoảng trắng, phần mở rộng là .C



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.4. Thực thi chương trình

✓ Run/Run (Ctrl+F9)



```
D:\BC\BIN\BC.EXE
Xin cho biet ten:Lan
Chao ban Lan
Chao mung ban den voi ngon ngu lap trinh C
```

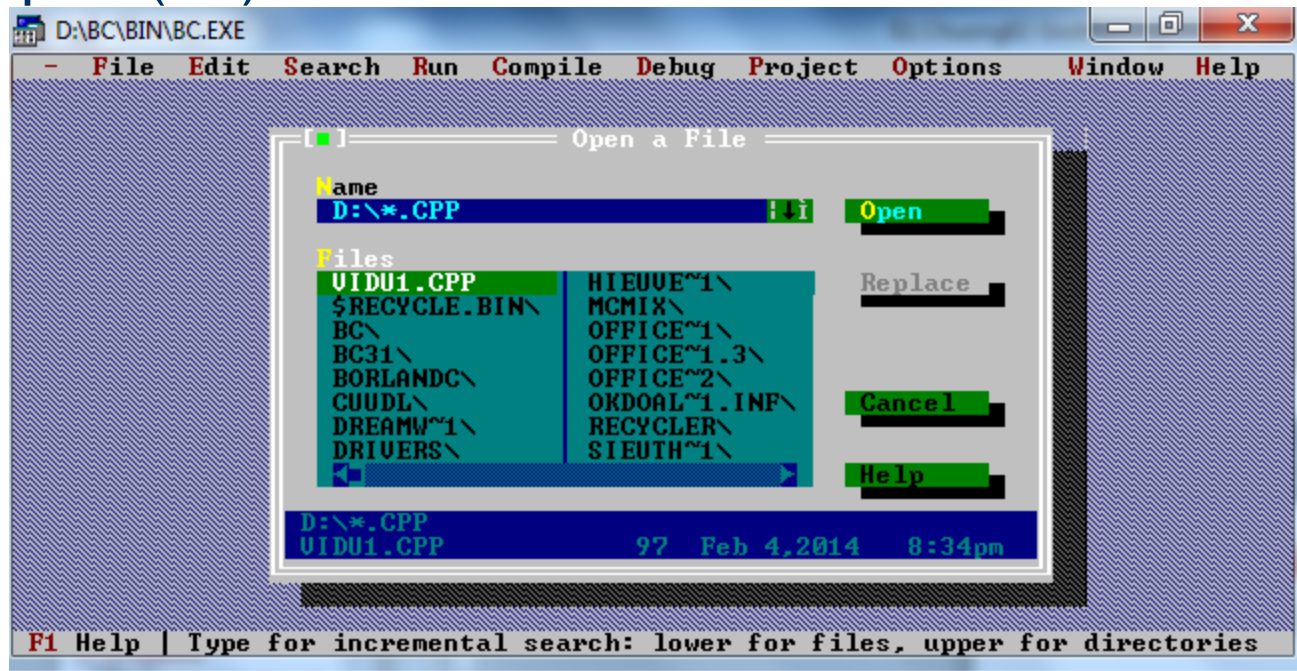
Nhập và Enter để kết thúc

Enter để kết thúc chương trình

2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.5. Mở chương trình đã có trên đĩa

✓ File/Open (F3)

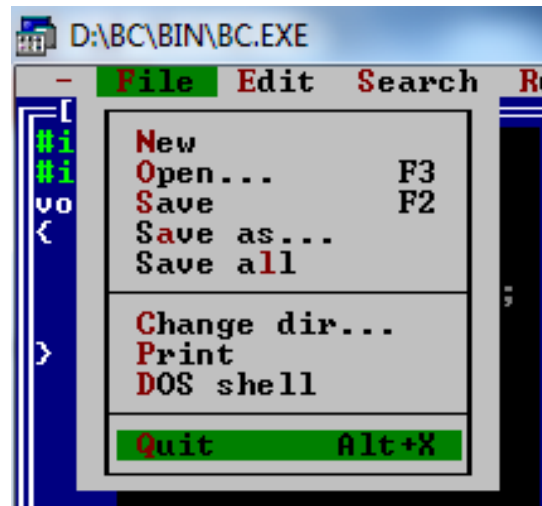
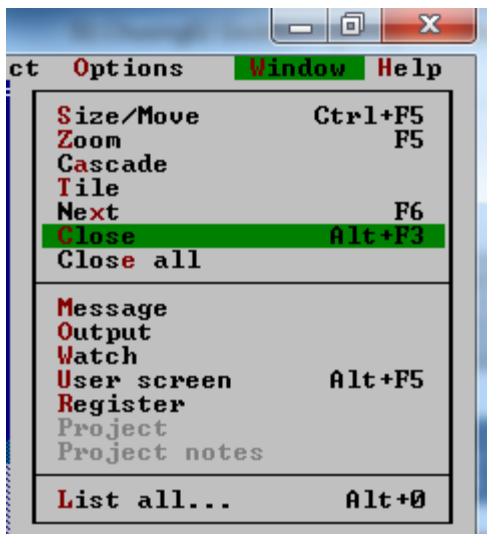


Gõ hoặc chọn tập tin cần mở → Open

2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.6. Thoát Borland C

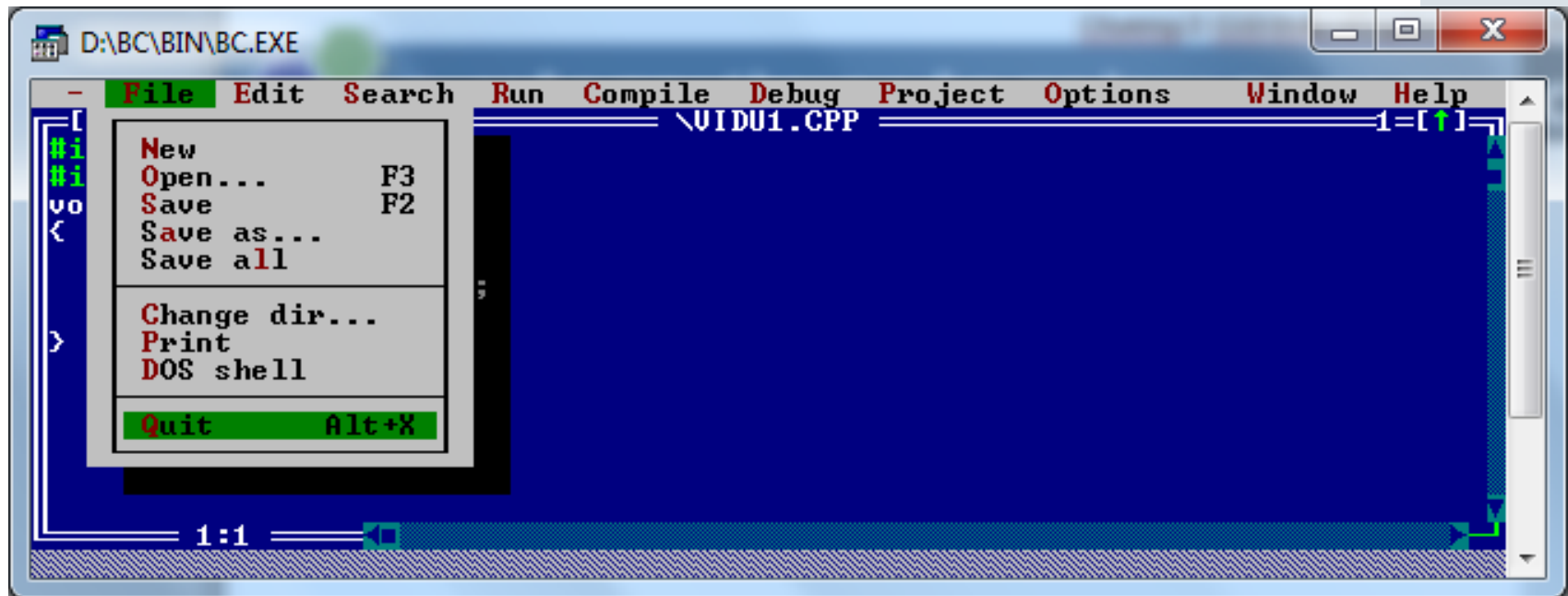
- ✓ Đóng 1 chương trình: Window/Colse (Alt + F3)
- ✓ Thoát Borland C: File/Exit (Alt + X)



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.7. Các lệnh trên thanh menu

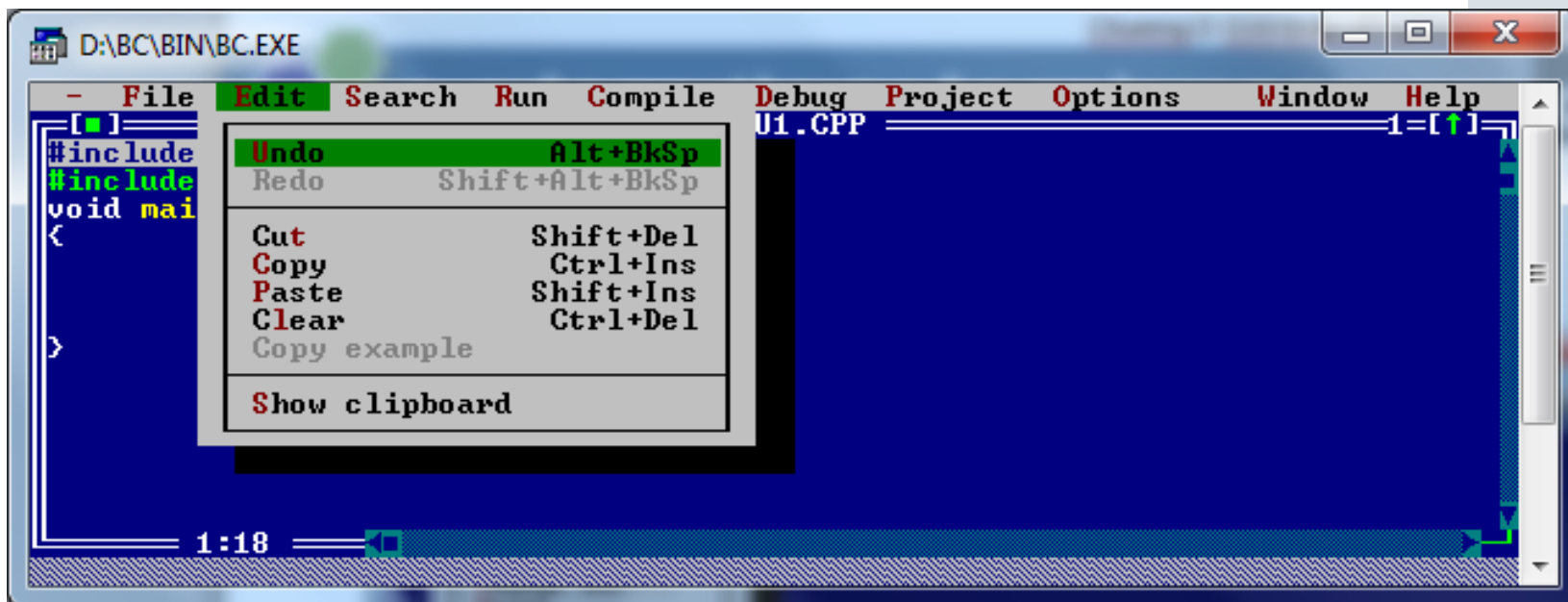
✓ Menu File



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.7. Các lệnh trên thanh menu(tt)

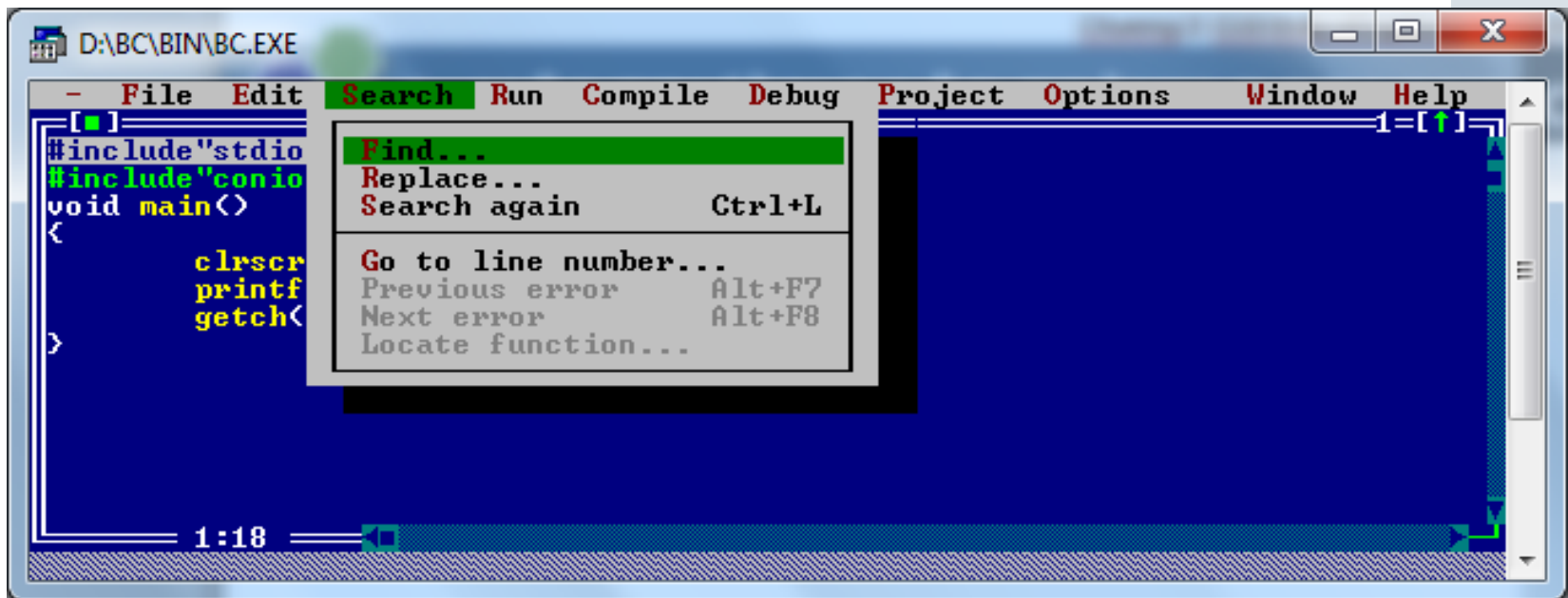
✓ Menu Edit



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.7. Các lệnh trên thanh menu(tt)

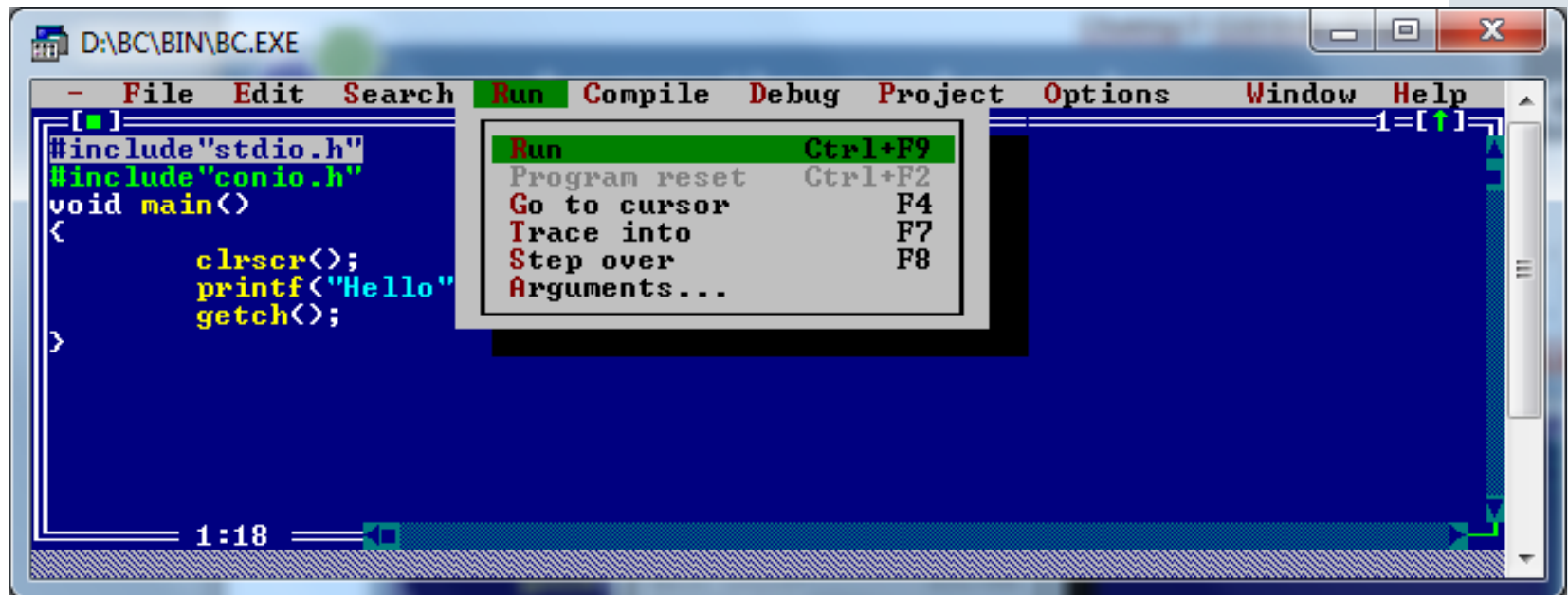
✓ Menu Search



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.7. Các lệnh trên thanh menu(tt)

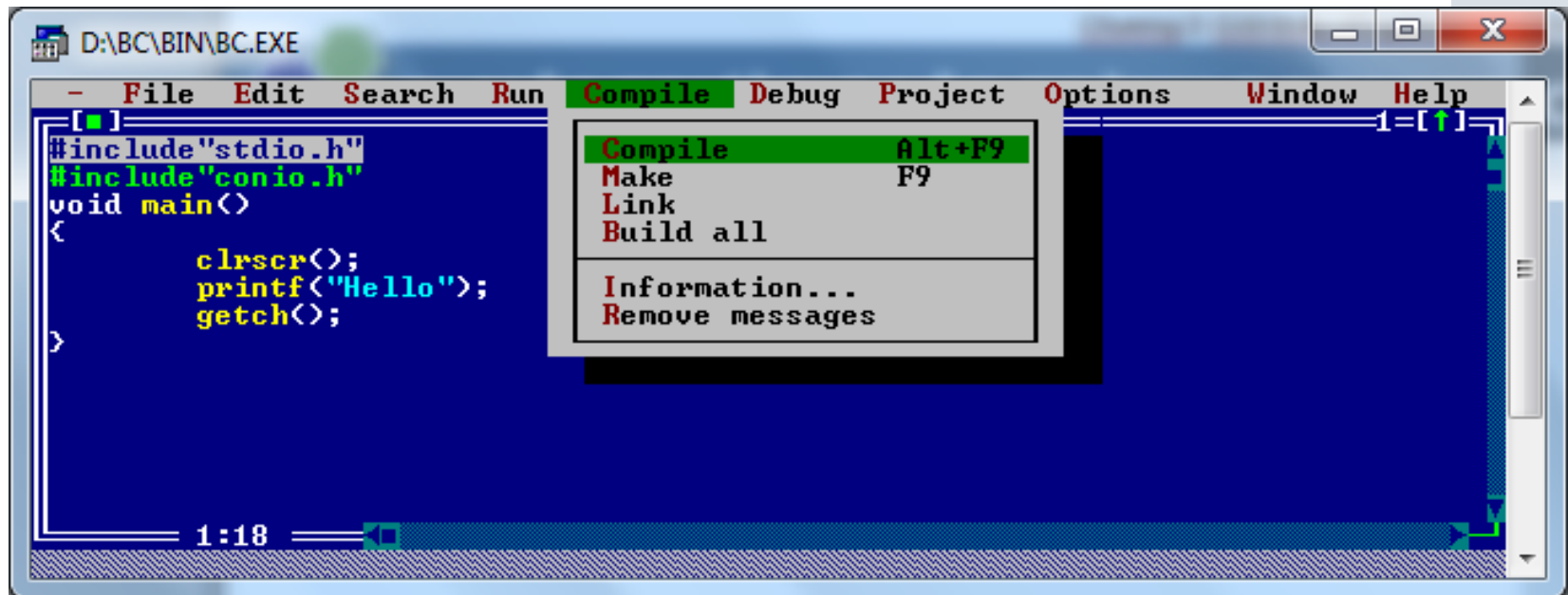
✓ Menu Run



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.7. Các lệnh trên thanh menu(tt)

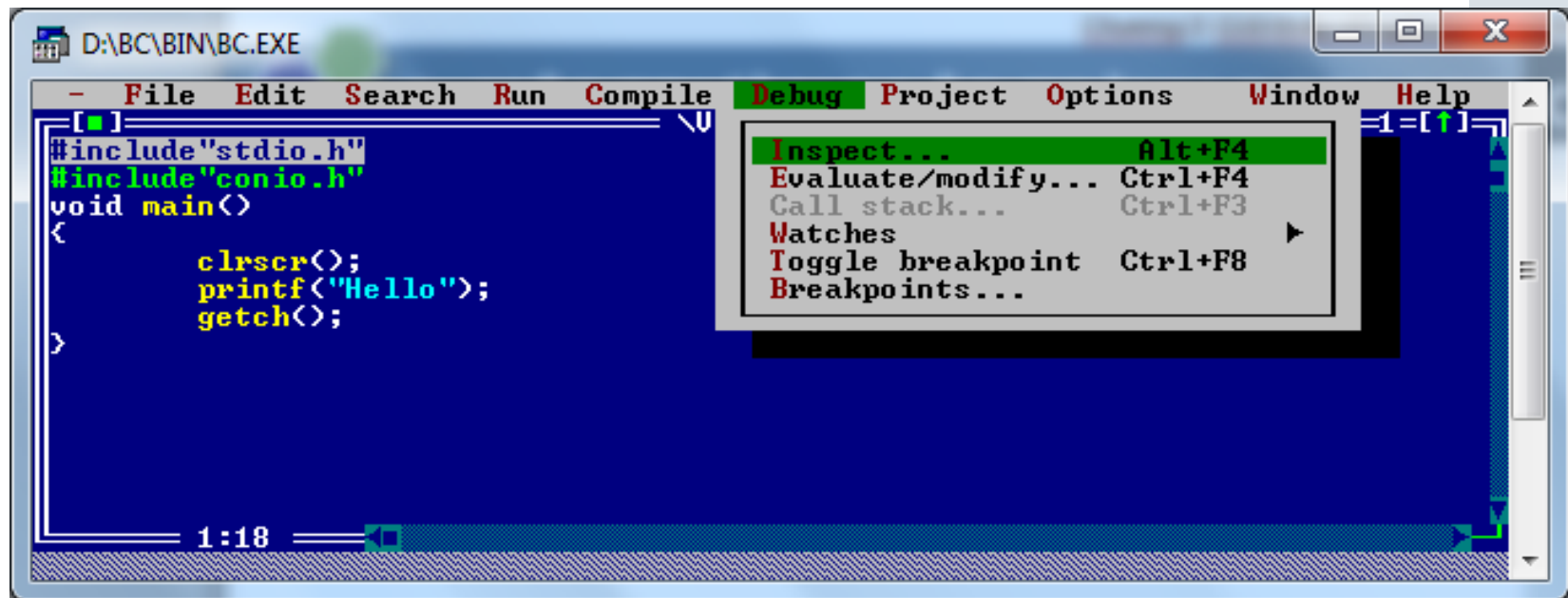
✓ Menu Compile



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.7. Các lệnh trên thanh menu(tt)

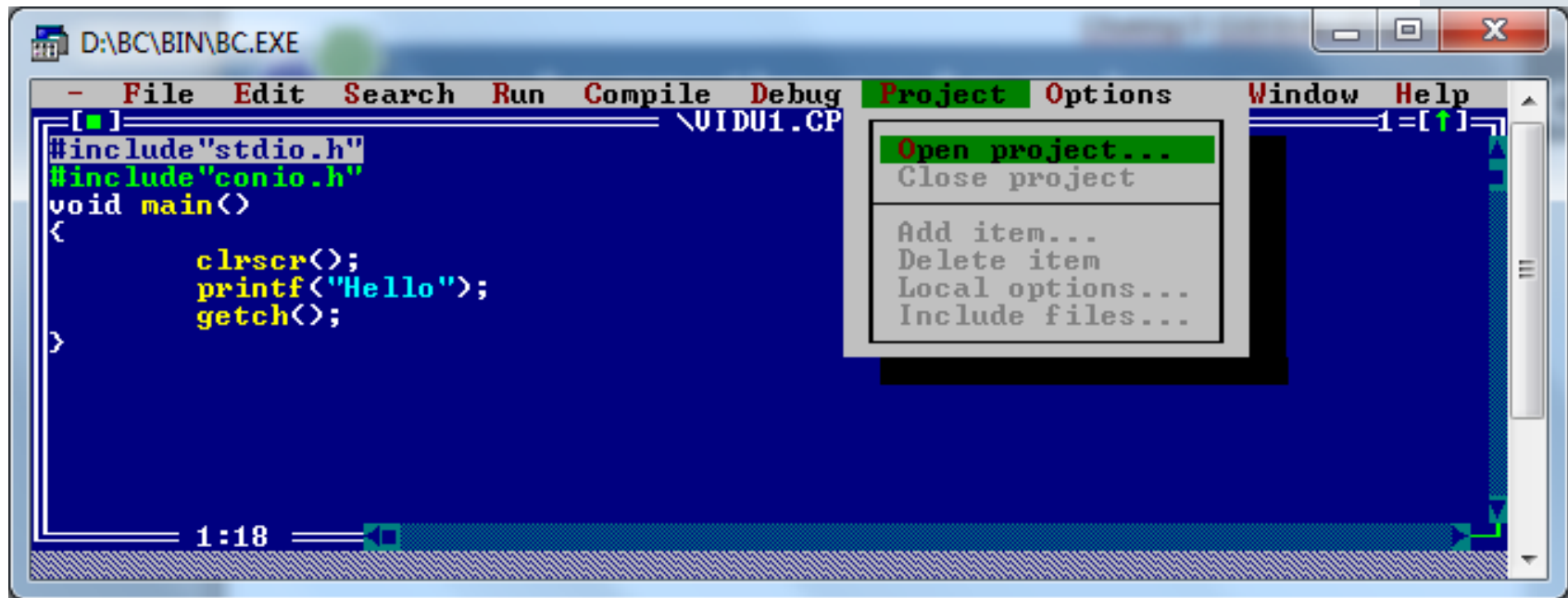
✓ Menu Debug



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.7. Các lệnh trên thanh menu(tt)

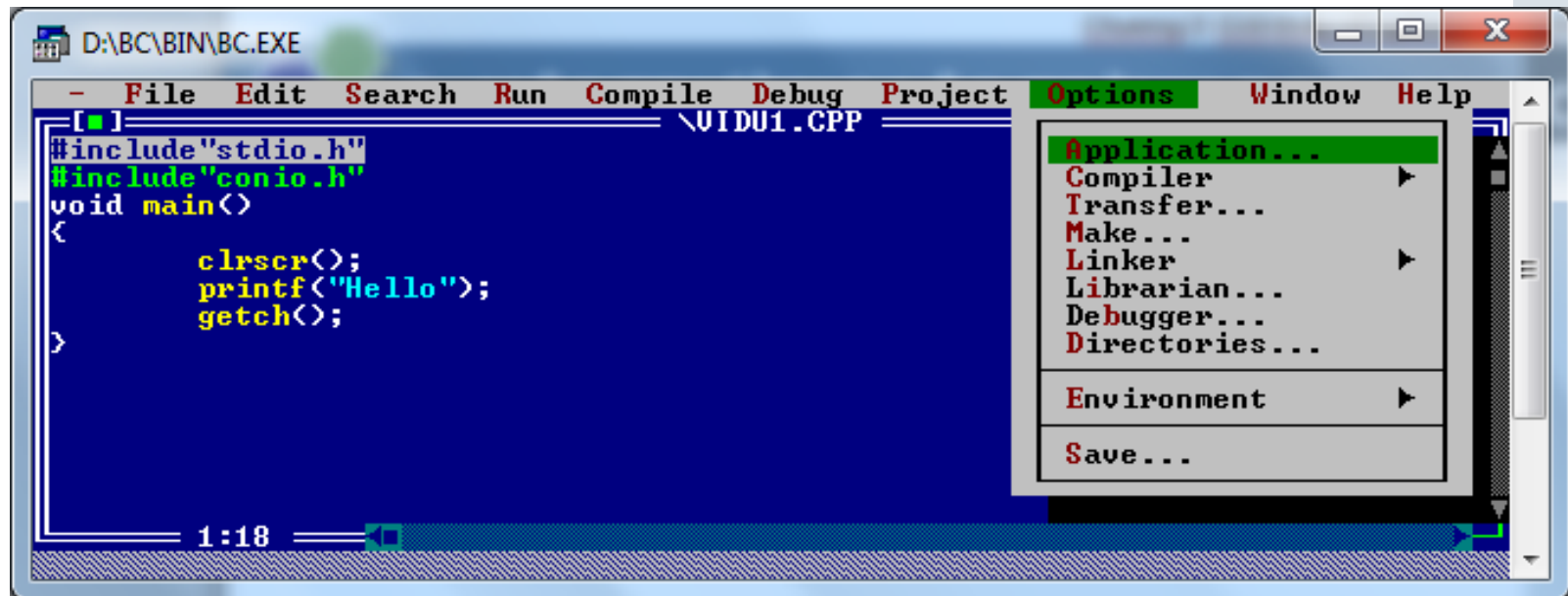
✓ Menu Project



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.7. Các lệnh trên thanh menu(tt)

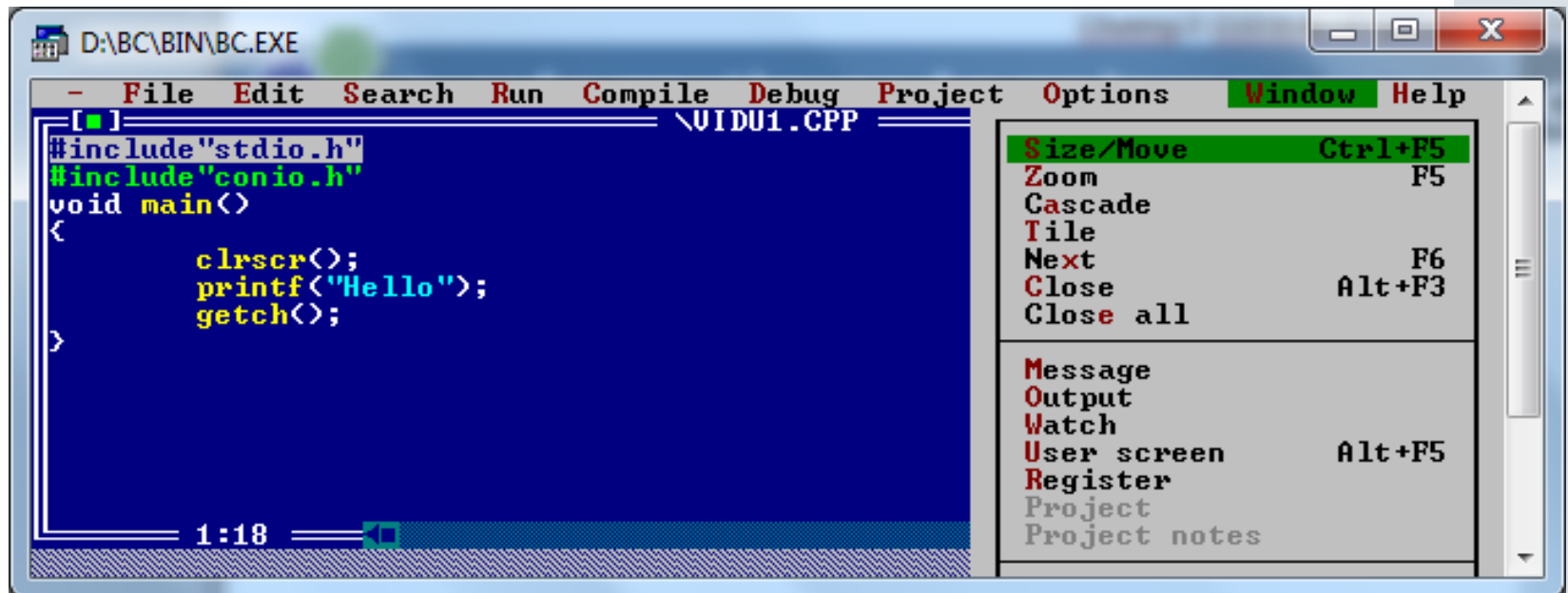
✓ Menu Option



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.7. Các lệnh trên thanh menu(tt)

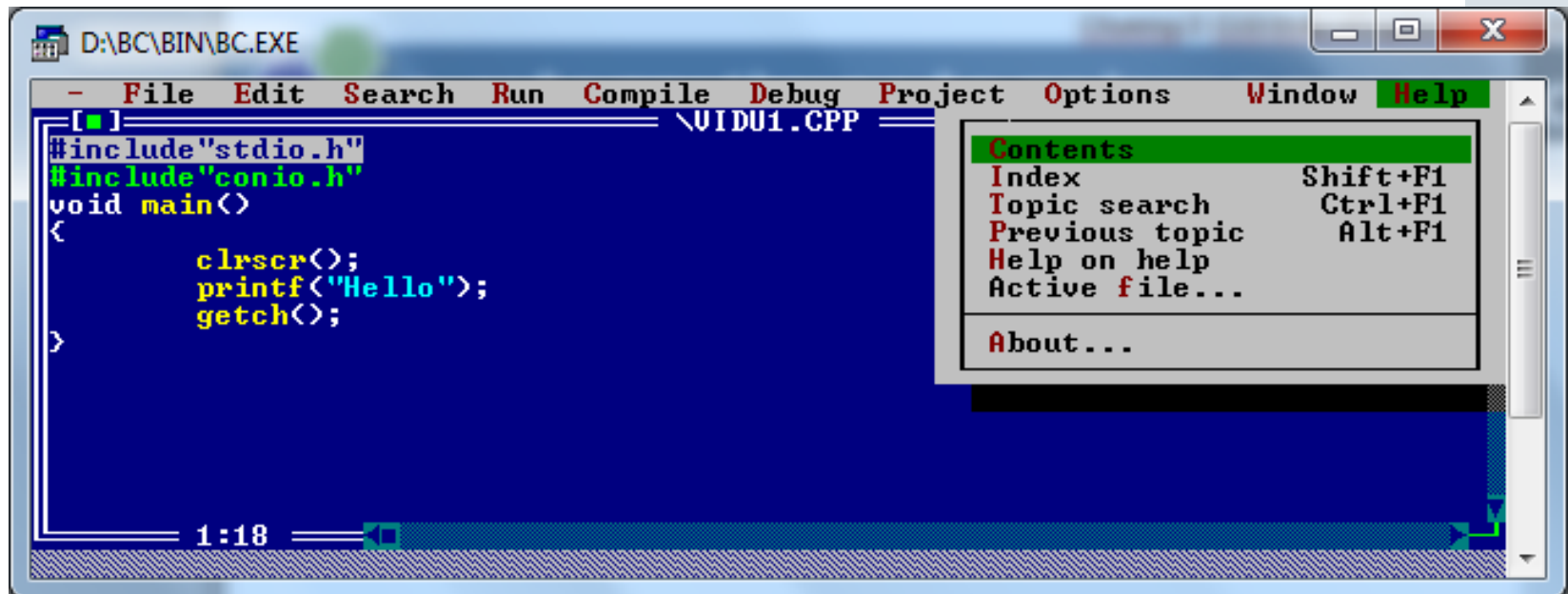
✓ Menu Window



2. MÔI TRƯỜNG LẬP TRÌNH BORLAND C

2.7. Các lệnh trên thanh menu(tt)

✓ Menu Help





3. CÁC CÂU LỆNH

3.1. Khái niệm câu lệnh

- ✓ Câu lệnh xác định 1 công việc mà chương trình phải thực hiện
- ✓ Mỗi câu lệnh được kết thúc bởi dấu chấm phẩy ;

3.2. Phân loại

- ✓ Lệnh đơn gồm lệnh gán, nhập, xuất
- ✓ Lệnh có cấu trúc là lệnh: Chứa các lệnh lệnh khác (lệnh điều kiện rẽ nhánh, lệnh điều kiện lựa chọn, lệnh lặp, . .
- ✓ Lệnh hợp thành (khối lệnh) là 1 nhóm lệnh, được đặt giữa cặp ngoặc { }





3 CÁC CÂU LỆNH

3.3. Lệnh gán

- ✓ Dùng để gán giá trị của 1 biểu thức cho biến
- ✓ Cú pháp:

$\text{<Tên biến> = <Biểu thức>}$

- ✓ Ví dụ:

```
void main()
{
    int x,y;
    x=10;
    y=2*x;
}
```

- ✓ Kiểu dữ liệu của biến phải cùng kiểu dữ liệu với biểu thức





3. CÁC CÂU LỆNH

3.3. Lệnh gán (tt)

Ghi chú:

- ✓ Trong 1 số trường hợp cần tương thích về kiểu dữ liệu (chuyển kiểu), Cú pháp:

(Tên kiểu) <Biểu thức>

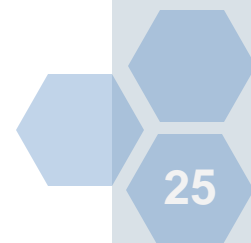
- ✓ Có thể gán giá trị cho biến khi khai báo

Ví dụ:

```
int x=10;
```

```
float f;
```

```
f = (float) x/4; /* f = 2.5 */
```





3. CÁC CÂU LỆNH

3.4. Lệnh nhập giá trị - scanf

- ✓ Hàm scanf nằm trong thư viện stdio.h
- ✓ Cú pháp:

scanf(“chỗi định dạng”, &<tên biến>)

Chuỗi định dạng: Quy định kiểu, độ rộng, . . .

Định dạng	Ý nghĩa
%[n]d	Nhập số nguyên tối đa n chữ số
%[n]f	Nhập số thực tối đa n chữ số
%c	Nhập 1 ký tự

Ví dụ:

%d → Nhập 1 số nguyên

%4d → Nhập số nguyên nhiều hơn 4 ký số, chỉ lấy 4

%6f → Nhập nhiều hơn 6 chỉ lấy 6 bao gồm dấu chấm



3. CÁC CÂU LỆNH

3.4. Lệnh nhập giá trị - scanf (tt)

Ví dụ:

```
scanf("%d", &bien1);
```

```
scanf("%f",&bien2);
```

```
scanf("%d%f%c",&bien1,&bien2,&bien3); //theo thứ tự
```

Ghi chú:

- ✓ Để nhập kiểu char chính xác dùng hàm fflush(stdin); để xóa các ký tự còn rong vùng đệm (giá trị biến trước bị cắt còn sót)
- ✓ Để nhập chuỗi ký tự (kết thúc là khoảng trắng) phải khai báo: char chuoi1[n] hoặc *chuoi2 và sử dụng %s để nhập.
- ✓ Để nhập chuỗi có chứa khoảng trắng (kết thúc enter) thì dùng hàm gets(chuoi1)



3. CÁC CÂU LỆNH

3.5. Lệnh xuất giá trị ra màn hình- printf

- ✓ Hàm printf nằm trong thư viện stdio.h
- ✓ Cú pháp:

printf("các chuỗi định dạng", <các biểu thức>);

Chuỗi định dạng: Quy định kiểu, cách biểu diễn, . . .

Định dạng	Ý nghĩa
%d	Xuất số nguyên
%e	Xuất số nguyên dạng khoa học (nhân 10 mũ x)
%[n]f	Xuất số thực n chữ số bao gồm dấu .
%[.n]f	Xuất số thực có n chữ số thập phân
%o	Xuất số nguyên hệ bát phân
%x	Xuất số nguyên hệ thập lục phân
%c	Xuất 1 ký tự
%s	Xuất chuỗi ký tự



3. CÁC CÂU LỆNH

3.5. Lệnh xuất giá trị ra màn hình- printf(tt)

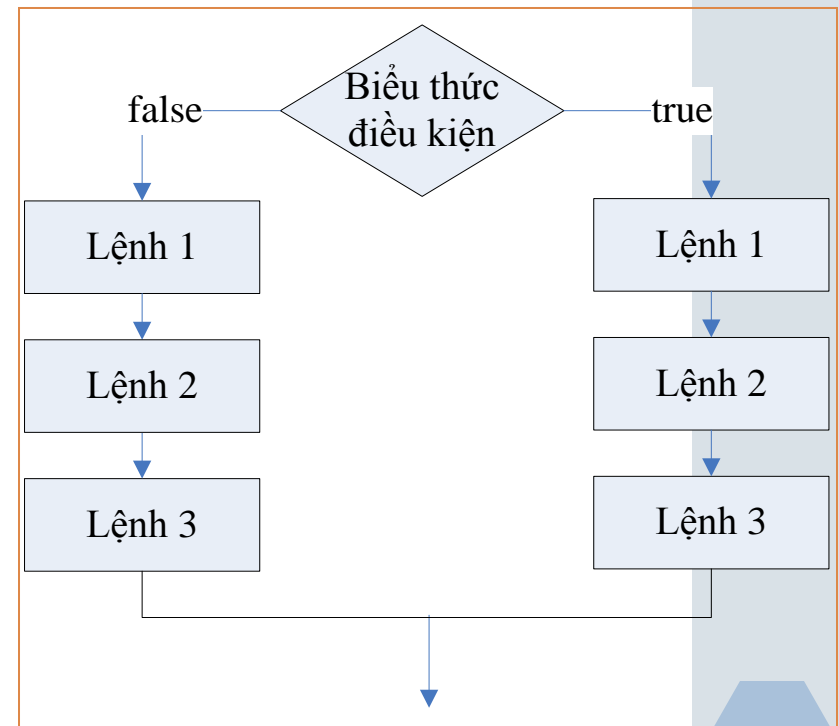
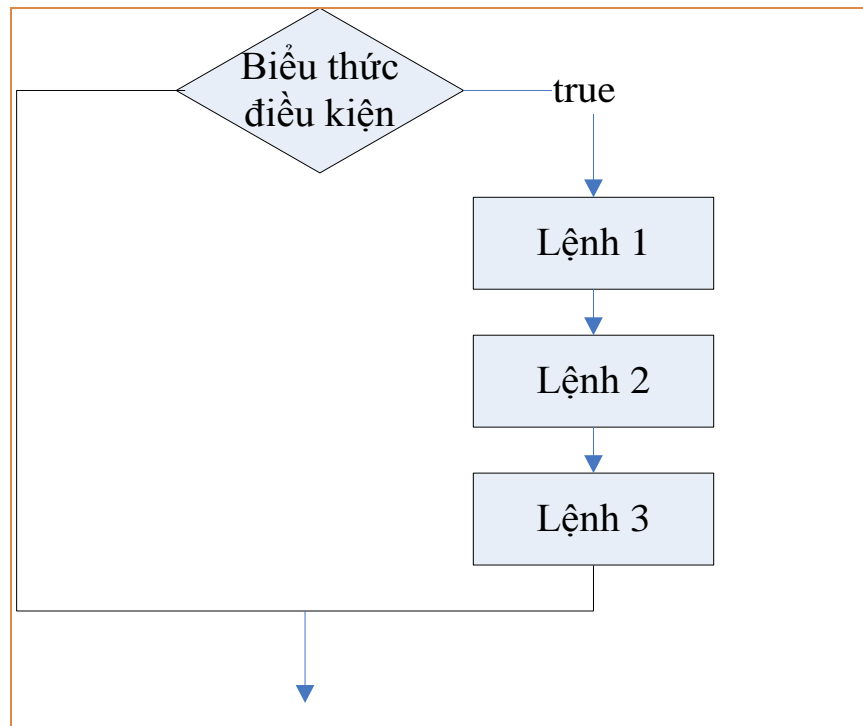
✓ Một số ký tự điều khiển:

Ký tự điều khiển	Ký tự hiển thị	Ý nghĩa
\f	FF	Sang trang
\n	LF	Xuống dòng
\r	CR	Về đầu dòng
\t	HT	Tab theo cột
\\	\	Dấu \
\'	'	Nháy đơn
\"	"	Nháy kép
\?	?	Dấu ?
..



4. CẤU TRÚC Rẽ NHÁNH

Cấu trúc rẽ nhánh cho phép thực hiện một khối lệnh dựa vào kết quả của một biểu thức điều kiện, gồm 2 dạng:





4. CẤU TRÚC Rẽ NHÁNH

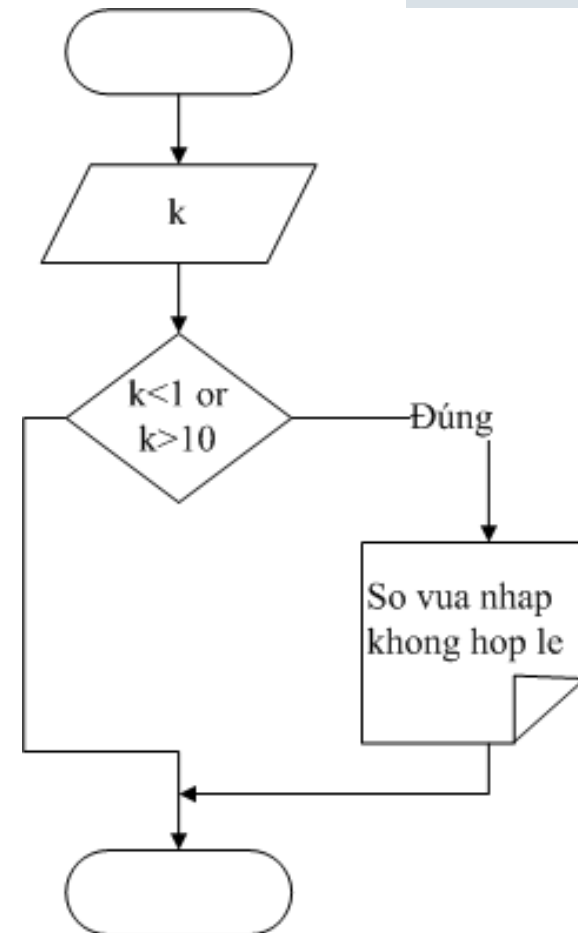
<pre>if (Biểu thức ĐK) khối lệnh 1; </pre>	<pre>if (Biểu thức ĐK) khối lệnh 1; else khối lệnh 2 ; </pre>
<p>Nếu biểu thức đúng thì sẽ thực hiện khối lệnh 1 và sau đó sẽ thực hiện các lệnh tiếp sau khối lệnh 1..</p> <p>Nếu biểu thức sai thì bỏ qua khối lệnh 1 mà thực hiện ngay các lệnh tiếp sau khối lệnh 1.</p>	<p>Nếu biểu thức đúng thì sẽ thực hiện khối lệnh 1 và sau đó sẽ thực hiện các lệnh tiếp sau khối lệnh 2.</p> <p>Nếu biểu thức sai thì bỏ qua khối lệnh 1 mà thực hiện khối lệnh 2 sau đó thực hiện tiếp các lệnh tiếp sau khối lệnh 2.</p>



4. CẤU TRÚC RỄ NHÁNH

Ví dụ 01: Viết chương trình nhập vào một số nguyên từ 1 đến 10, nếu nhập sai thì thông báo

```
void main()
{
    int k;
    printf("Hay nhap mot so tu 1 den 10: ");
    scanf("%d", &k);
    if (k < 1 || k > 10)
        printf<<"So nhap khong hop le ";
    getch();
}
```

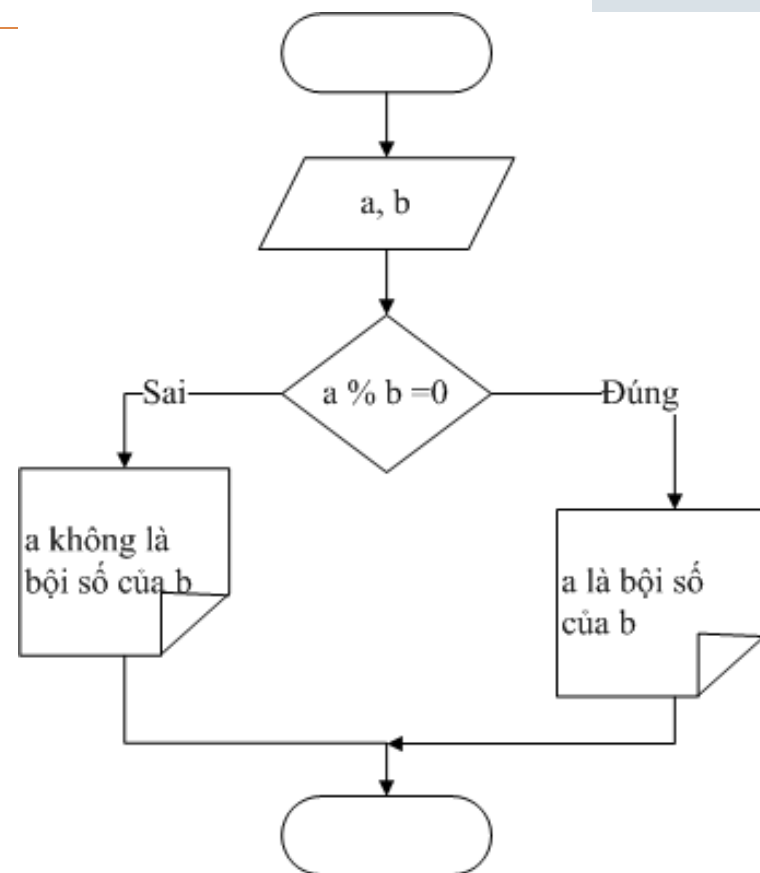




4. CẤU TRÚC RỄ NHÁNH

Ví dụ 02: Nhập vào số nguyên a và b, nếu a là bội số của b thì in thông báo “a là bội số của b”, ngược lại in “a không là bội số của b”

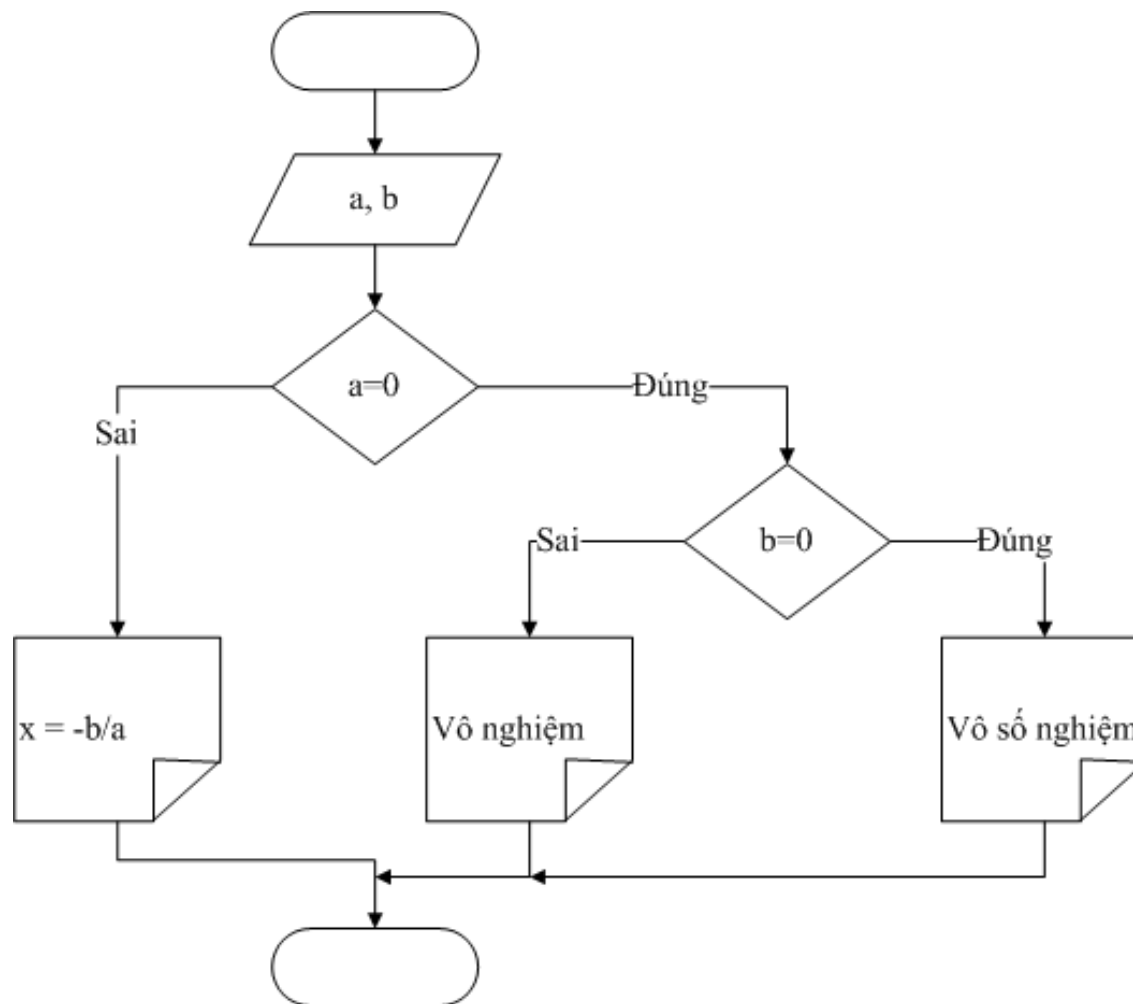
```
void main()
{
    int a, b;
    printf("Nhap vao a: ");
    scanf("%d",a);
    printf("Nhap vao b: ");
    scanf("%d",b);
    if(a%b==0)
        printf("a la boi so cua b");
    else
        printf("a khong la boi so cua b");
    getch();
}
```





4. CẤU TRÚC Rẽ NHÁNH

Ví dụ 03: Giải và biện luận phương trình: $ax+b=0$





4. CẤU TRÚC RỄ NHÁNH

Ví dụ 03: Giải và biện luận phương trình: $ax+b=0$

```
void main()
{
    float a, b;
    printf("Nhap vao a: ");          scanf("%f",&a);
    printf("Nhap vao b: ");          scanf("%f",&b);
    if (a == 0)
        if (b == 0)
            printf("Phuong trinh vo so nghiem");
        else
            printf("Phuong trinh vo nghiem");
    else
        cout<<"PT co nghiem x =%.2f", -b / a);
    getch();
}
```



5. CẤU TRÚC LỰA CHỌN

```
switch (biểu thức)
{
    case n1:
        các câu lệnh ;
        break ;
    case n2:
        các câu lệnh ;
        break ;

    .....
    case nk:
        <các câu lệnh> ;
        break ;
    [default: các câu lệnh]
}
```



5. CẤU TRÚC LỰA CHỌN

- ✓ n_i là các hằng số nguyên hoặc ký tự.
- ✓ Phụ thuộc vào giá trị của biểu thức sau switch, nếu:
 - Giá trị này = n_i thì thực hiện câu lệnh sau case n_i .
 - Khi giá trị biểu thức không thỏa tất cả các n_i thì thực hiện câu lệnh sau **default** nếu có, hoặc thoát khỏi câu lệnh **switch**.
 - Khi chương trình đã thực hiện xong câu lệnh của **case** n_i thì sẽ thực hiện luôn các lệnh thuộc **case** bên dưới mà không xét lại điều kiện → Vậy, để chương trình thoát khỏi lệnh **switch** sau khi thực hiện xong một trường hợp, ta dùng lệnh **break**.



5. CẤU TRÚC LỰA CHỌN

Ví dụ: Nhập vào số nguyên n có giá trị từ 1 đến 5. In cách đọc của số đó ra màn hình.

```
void main()
{
    int n;
    printf("Nhap vao n (1<=n<=5) ");    scanf("%d", &n);
    switch (n)
    {
        case 1: printf("So mot"); break;
        case 2: printf("So hai"); break;
        case 3: printf("So ba"); break;
        case 4: printf("So bon"); break;
        case 5: printf("So nam"); break;
        default : printf("Khong doc duoc");
    }
    getch();
}
```



6. CẤU TRÚC LẶP

- ✓ Cấu trúc lặp For
- ✓ Cấu trúc lặp While . . .
- ✓ Cấu trúc lặp Do . . . While



6.1 CẤU TRÚC LẶP FOR

```
for (<BT khởi gán> ; <BT điều kiện>; <BT tăng/giảm>)  
{  
    <khối lệnh>;  
}
```

- *Bước 1: Thực hiện Biểu thức khởi gán*
- *Bước 2: Kiểm tra điều kiện của biểu thức điều kiện.*
Nếu biểu thức điều kiện bằng true thì cho thực hiện các lệnh trong vòng lặp, thực hiện biểu thức tăng giảm. Quay trở lại bước 2. Ngược lại thoát khỏi lặp.

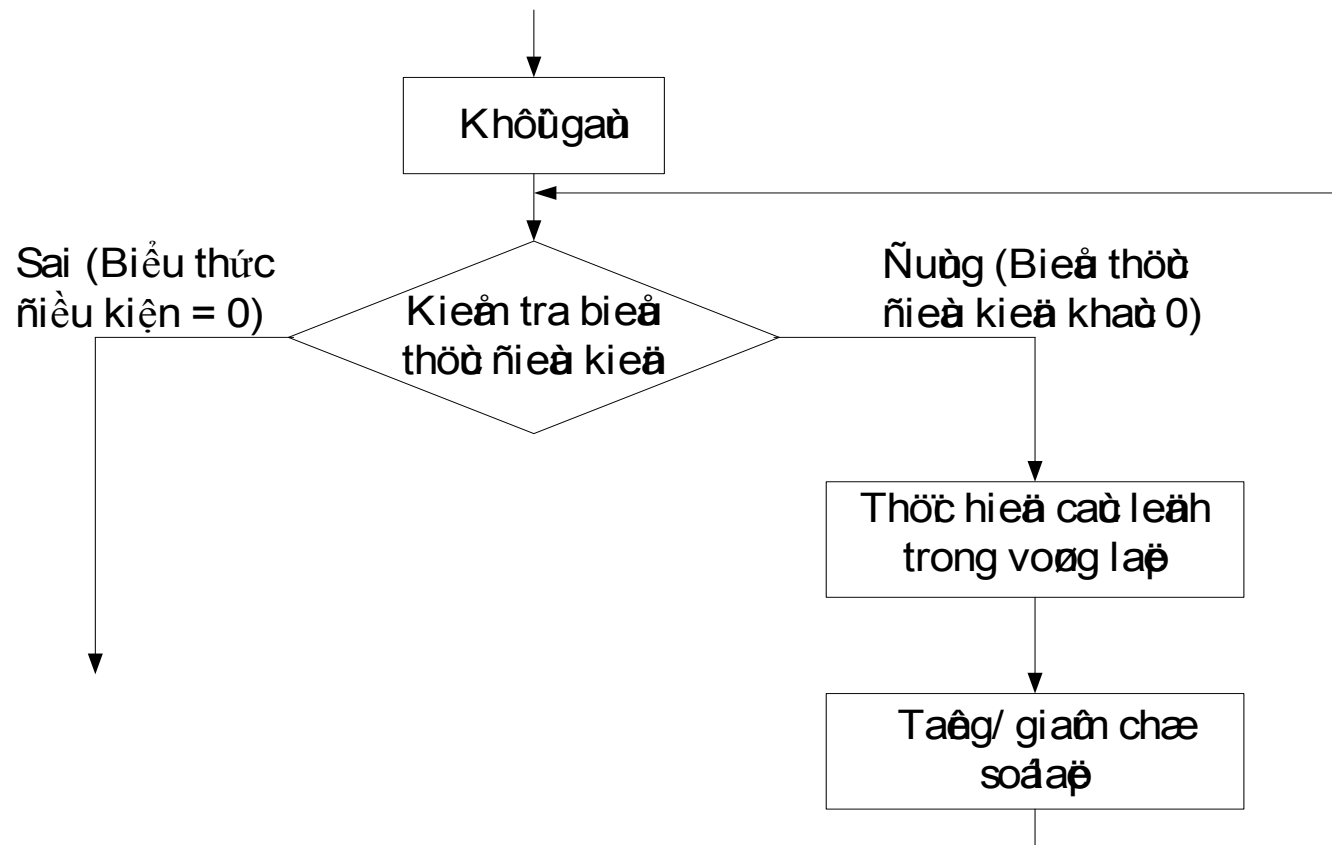
Bất kỳ biểu thức nào trong 3 biểu thức nói trên đều có thể vắng nhưng phải giữ dấu chấm phẩy (;)





6.1 CẤU TRÚC LẶP FOR

Nguyên lý hoạt động

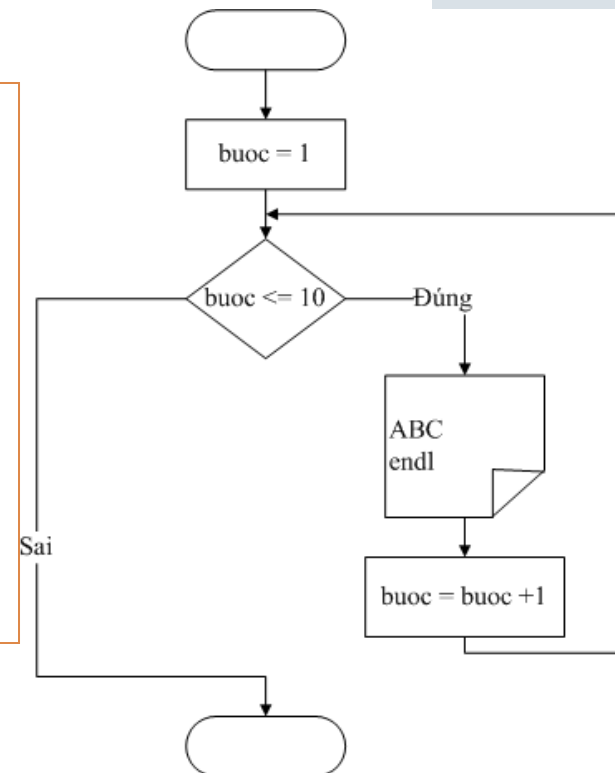




6.1 CẤU TRÚC LẶP FOR

Ví dụ : In ra màn hình 10 dòng chữ “*Xin chao*”

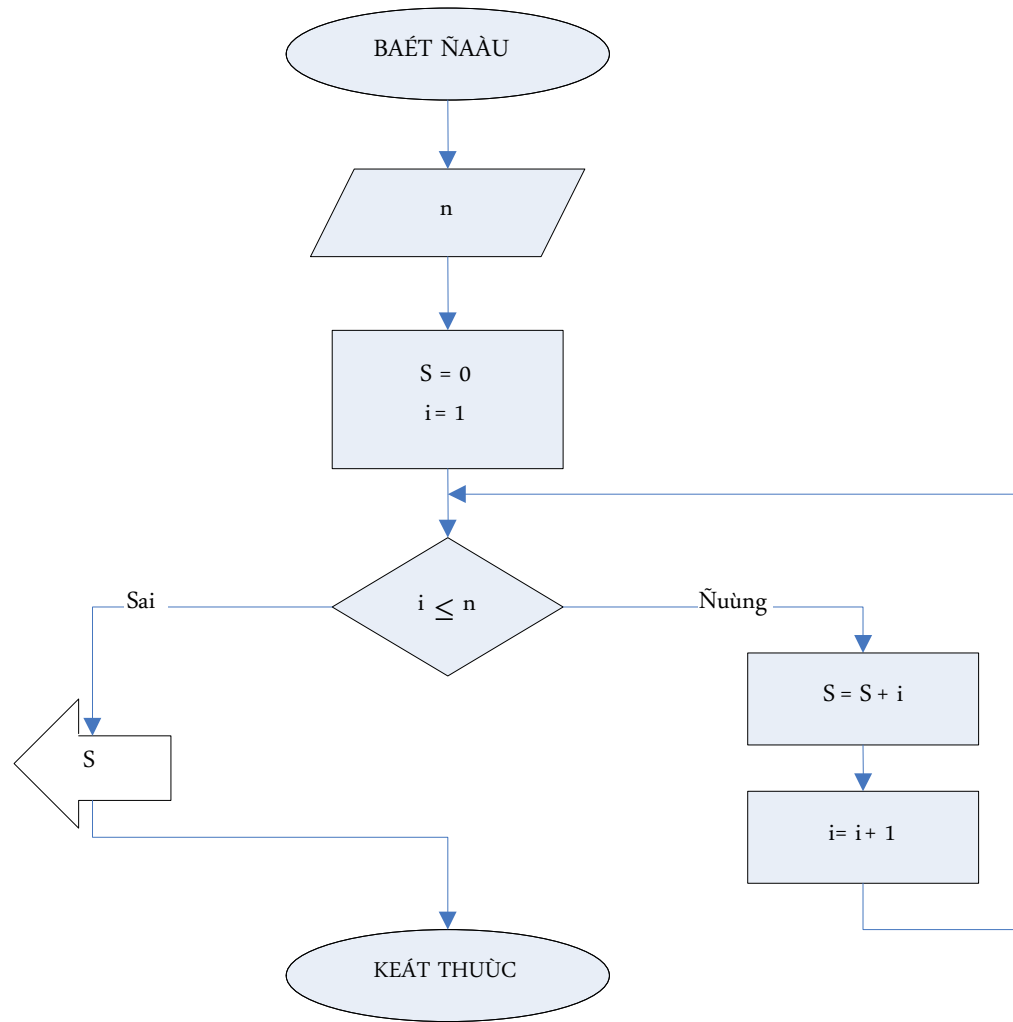
```
void main()
{
    for (int buoc = 1; buoc <= 10; buoc++)
        printf("Xin chao");
    getch();
}
```





6.1. CẤU TRÚC LẶP FOR

Ví dụ: Tính tổng: $S=1+2+3+ \dots +n$, với $n>0$





6.2 CẤU TRÚC LẶP WHILE

```
while ( <biểu thức điều kiện>)
```

```
{
```

```
    lệnh/ khối lệnh;
```

```
}
```

Kiểm tra điều kiện của biểu thức điều kiện. Nếu biểu thức điều kiện bằng true thì thực hiện các lệnh trong vòng lặp, thực hiện biểu thức tăng giảm. Quay trở lại bước 2. Ngược lại thoát khỏi lặp.



6.2 CẤU TRÚC LẶP WHILE

Ví dụ: In ra màn hình 10 dòng chữ “*Xin chào*”

```
void main()
{
    int buoc = 1;
    while(buoc <= 10)
    {
        printf("Xin chào");
        buoc++;
    }
}
```



6.3 CẤU TRÚC LẶP DO . . . WHILE

```
do  
{  
    <khối lệnh>;  
} while (biểu thức điều kiện);
```

- ✓ Thực hiện khối lệnh cho đến khi biểu thức có giá trị bằng false.
- ✓ Ngược lại với cấu trúc lặp while (kiểm tra điều kiện trước khi thực hiện lặp), vòng lặp do...while thực hiện lệnh lặp rồi mới kiểm tra điều kiện. Do đó vòng lặp do...while thực hiện lệnh ít nhất một lần.



6.3 CẤU TRÚC LẶP DO . . . WHILE

Ví dụ 08: Nhập vào một số nguyên dương, nếu nhập vào số bằng hoặc nhỏ hơn 0 thì thông báo lỗi và yêu cầu nhập lại.

```
void main()
{
    int n;
    do {
        printf("Nhap vao mot so nguyen duong: ");
        scanf("%d",&n);
        if (n <= 0)
            printf("Nhap sai, hay nhap lai! ");
    } while (n <= 0);
    printf("Ban da nhap dung, ket thuc chuong trinh");
}
```



7. CÁC CÂU LỆNH ĐẶC BIỆT

7.1. Lệnh Break:

- ✓ Thoát khỏi các cấu trúc *switch*, *while*, *for*, *do...while* chứa nó gần nhất.
- ✓ Tại thời điểm break không cần kiểm tra kết quả của biểu thức điều kiện.
- ✓ Tuy nhiên, cần phân biệt với lệnh return là lệnh trả về từ hàm, nghĩa là thoát khỏi hàm đang thi hành, nên cũng giúp thoát luôn khỏi tất cả các vòng lặp.



7. CÁC CÂU LỆNH ĐẶC BIỆT

7.2. Lệnh Continue:

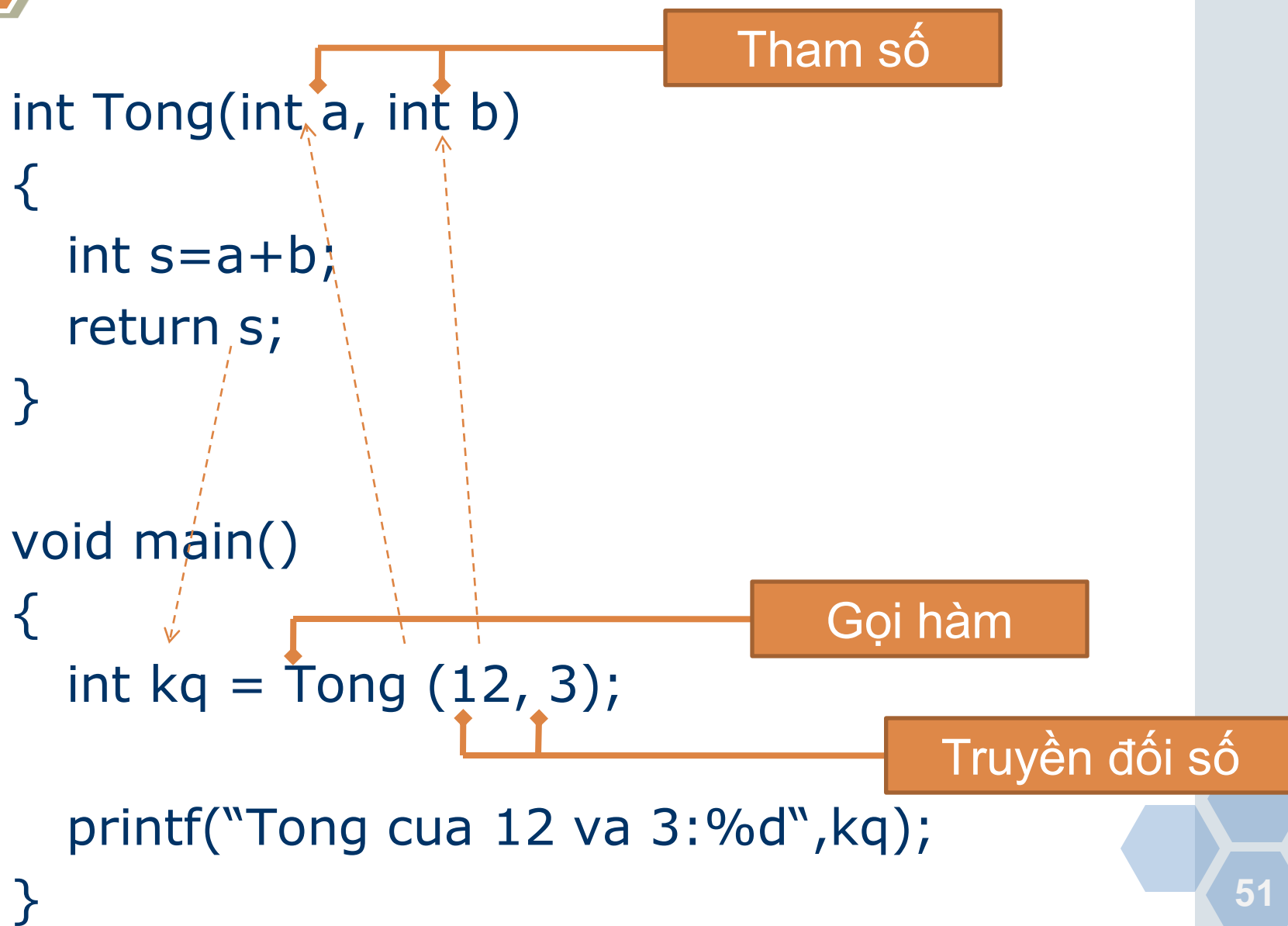
- ✓ Được sử dụng trong các vòng lặp như *while*, *for*, *do...while*.
- ✓ Khi lệnh *continue* được gọi thì chương trình sẽ quay trở về đầu vòng lặp để bắt đầu lần lặp mới (có kiểm tra điều kiện lặp để xác định có lặp tiếp hay không).
- ✓ Nếu có các lệnh còn lại (cùng trong vòng lặp) đặt sau *continue* sẽ không được thực hiện.
- ✓ Nói tóm lại, lệnh *continue* thường dùng để bỏ qua một lần lặp nào đó nếu thỏa điều kiện.



8. HÀM TRONG C

- ✓ **Hàm-Function** là một đoạn chương trình độc lập thực hiện trọn vẹn một công việc nhất định sau đó trả về giá trị cho chương trình gọi nó, hay nói cách khác hàm là sự chia nhỏ của chương trình.
- ✓ **Mục đích sử dụng hàm:**
 - Khi có một công việc giống nhau cần thực hiện ở nhiều vị trí.
 - Khi cần chia một chương trình lớn phức tạp thành các đơn thể nhỏ (hàm con) để chương trình được trong sáng, dễ hiểu trong việc xử lý, quản lý việc tính toán và giải quyết vấn đề.

8. HÀM TRONG C





8. HÀM TRONG C

✓ Định nghĩa hàm

```
Type tên hàm (Các tham số) {
    Khai báo các biến cục bộ;
    Các câu lệnh;
    [return [biểu thức];]
}
```

Thân hàm

Type: Kiểu dữ liệu giá trị trả về của hàm

Kiểu dữ liệu trả về của hàm gồm 2 loại

- **void:** Không trả về giá trị
- **float / int / long / char */ kiểu cấu trúc / ... :** Trả về giá trị kết quả có kiểu dữ liệu tương ứng



8. HÀM TRONG C

- ✓ **TênHàm:** Đặt tên ngắn gọn phản ánh đúng chức năng thực hiện của hàm
- ✓ **Danh sách Các tham số (nếu có):** Đầu vào của hàm (*trong một số trường hợp có thể là đầu vào và đầu ra của hàm nếu kết quả đầu ra có nhiều giá trị* - Tham số này gọi là tham chiếu)
- ✓ **Thân hàm :** Bắt đầu và kết thúc bằng các dấu { }. Trong thân hàm chứa các câu lệnh cần thiết để thực hiện một yêu cầu nào đó đã đề ra cho hàm.
Thân hàm có thể sử dụng một câu lệnh return, giá trị của biểu thức trong câu lệnh return sẽ được gán cho hàm.

8. HÀM TRONG C

✓ Sử dụng hàm

```
int Tong(int a, int b)
{
    int s=a+b;
    return s;
}
```

Tham số

```
void main()
{
```

```
    int kq = Tong (12, 3);
```

Gọi hàm

```
    cout<<"Tong cua 12 va 3: "<<kq;
```

Truyền đối số



8. HÀM TRONG C

- ✓ **Tham số là tham trị**
 - ✓ Tham số chứa dữ liệu đầu vào

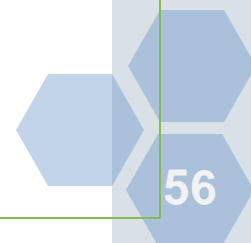
- ✓ **Tham số là tham chiếu (tham biến)**
 - ✓ Tham số làm kết quả đầu ra
 - ✓ Tham số vừa làm đầu vào và đầu ra
 - ✓ Dùng dấu & phía trước tên tham số khi cài đặt hàm



8. HÀM TRONG C

✓ Tham số là tham trị

```
void HoanVi(int a, int b){  
    int tam = a;  
    a = b;  
    b = tam;  
    printf("Trong HoanVi: a = %d; b=%d",a,b);  
}  
void main(){  
    int a = 5, b = 21;  
    printf("Truoc khi HoanVi: a =%d, b=%d ",a,b);  
    HoanVi(a, b);  
    printf("Sau khi goi HoanVi: a =%d, b=%d»,a,b);  
}
```





8. HÀM TRONG C

✓ Tham số là tham biến

```
void HoanVi(int &a, int &b){  
    int tam = a;  
    a = b;  
    b = tam;  
    printf("Trong HoanVi: a = %d; b=%d",a,b);  
}  
void main(){  
    int a = 5, b = 21;  
    printf("Truoc khi HoanVi: a =%d, b=%d ",a,b);  
    HoanVi(a, b);  
    printf("Sau khi goi HoanVi: a =%d, b=%d»,a,b);  
}
```