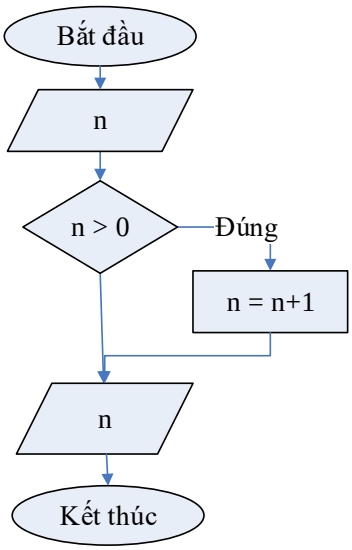
TÓM TẮT LÝ THUYẾT - PHẦN CƠ BẢN

I. Lưu đồ giải thuật

1) Cấu trúc lựa chọn

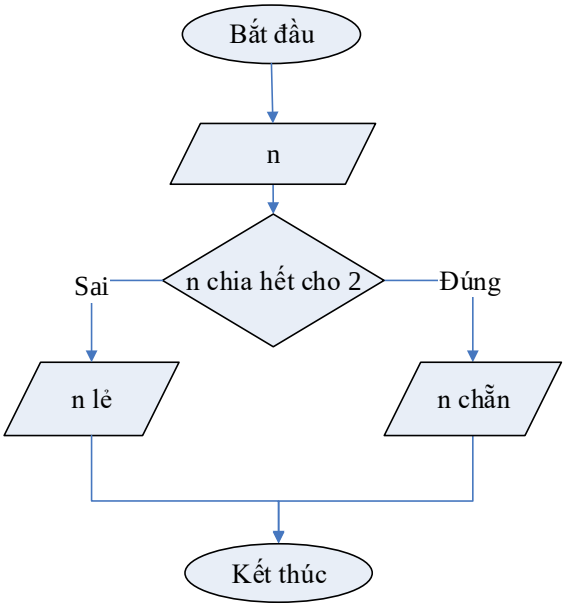
(a) Chỉ xét trường hợp khi điều kiện đúng

Ví dụ: Nhập vào số nguyên n. Kiểm tra nếu n > 0 thì tăng n lên 1 đơn vị. Xuất kết quả.



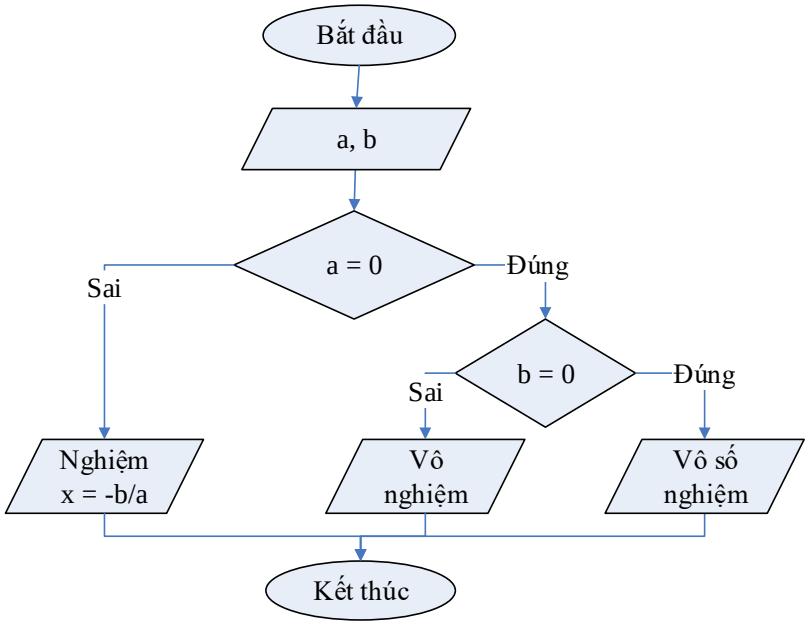
(b) Xét cả hai trường hợp đúng hoặc sai

Ví dụ: Nhập vào số nguyên n. Kiểm tra nếu n chẵn xuất ra màn hình "n chẵn", ngược lại xuất "n lẻ".



(c) **Xét nhiều trường hợp**

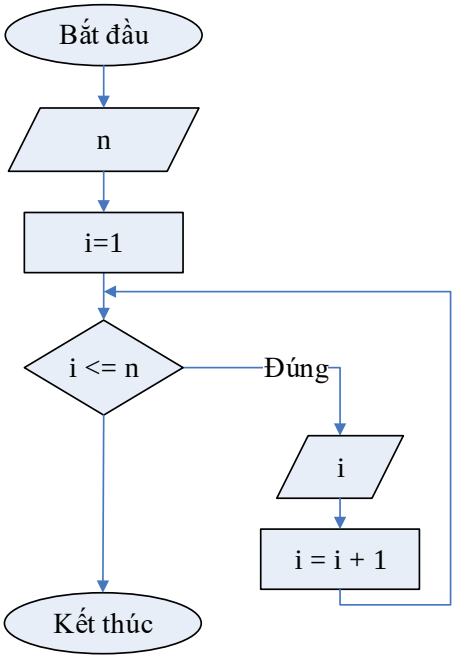
Ví dụ: Giải và biện luận phương trình bậc nhất: ax + b =0.



2) Cấu trúc lặp: Thực hiện liên tục 1 lệnh hay tập lệnh với số lần lặp dựa vào điều kiện. Quá trình lặp sẽ kết thúc khi điều kiện bị vi phạm.

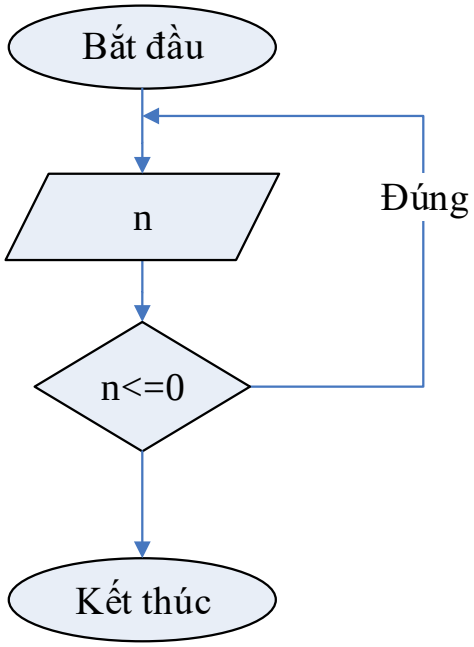
(a) **Kiểm tra điều kiện trước khi lặp**

Ví dụ: Nhập vào số nguyên n. Xuất ra màn hình từ 1 đến n.



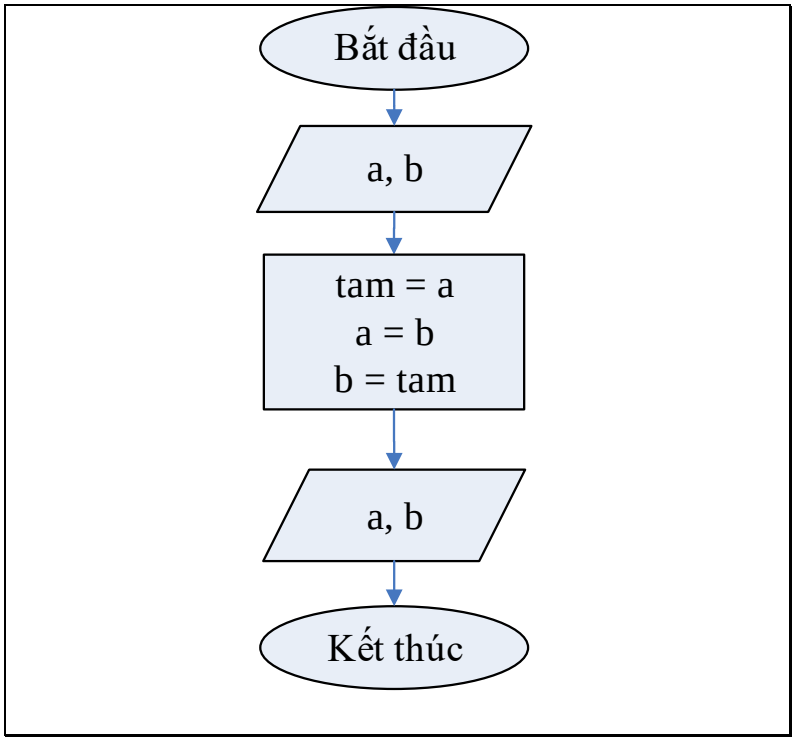
(b) **Thực hiện lặp trước khi kiểm tra điều kiện**

Ví dụ: Nhập vào số nguyên dương n. Nếu nhập sai yêu cầu nhập lại n.

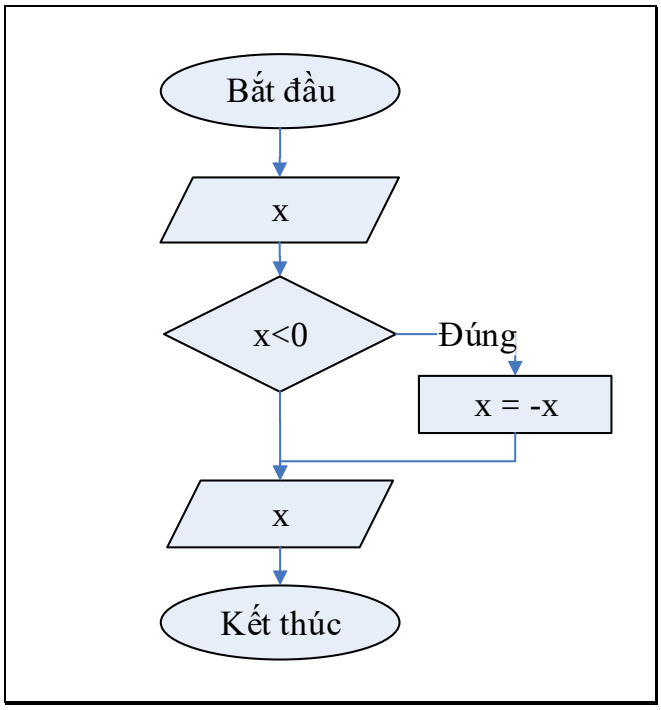


3) Một số ví dụ

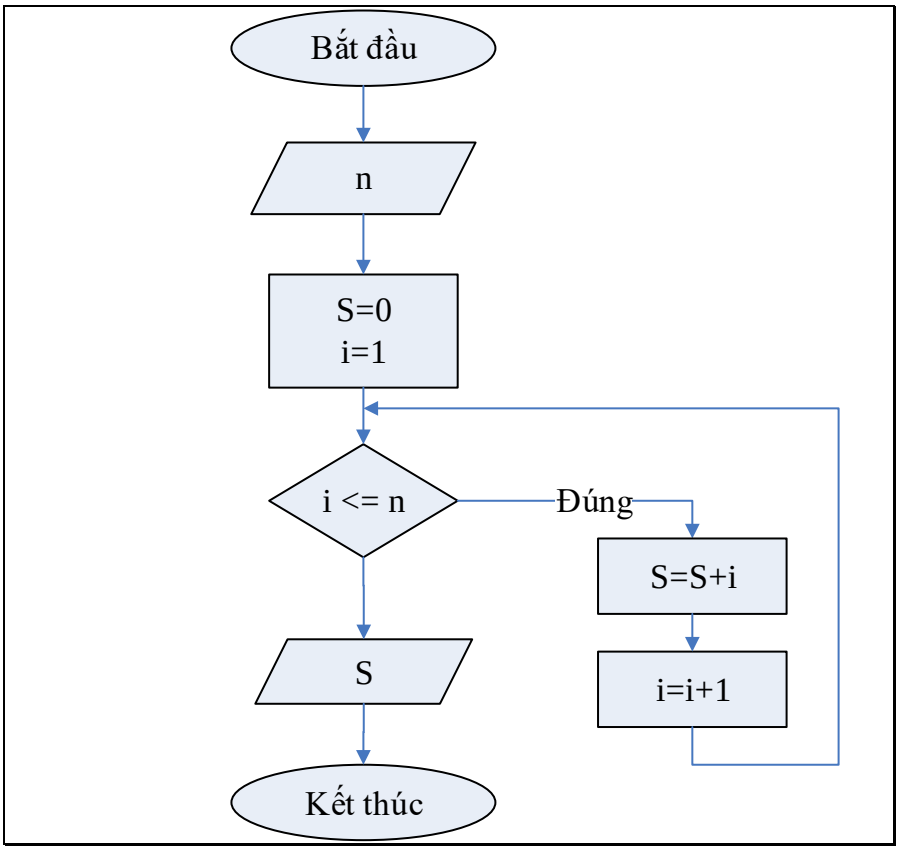
Ví dụ 1: Hoán vị hai số nguyên a và b



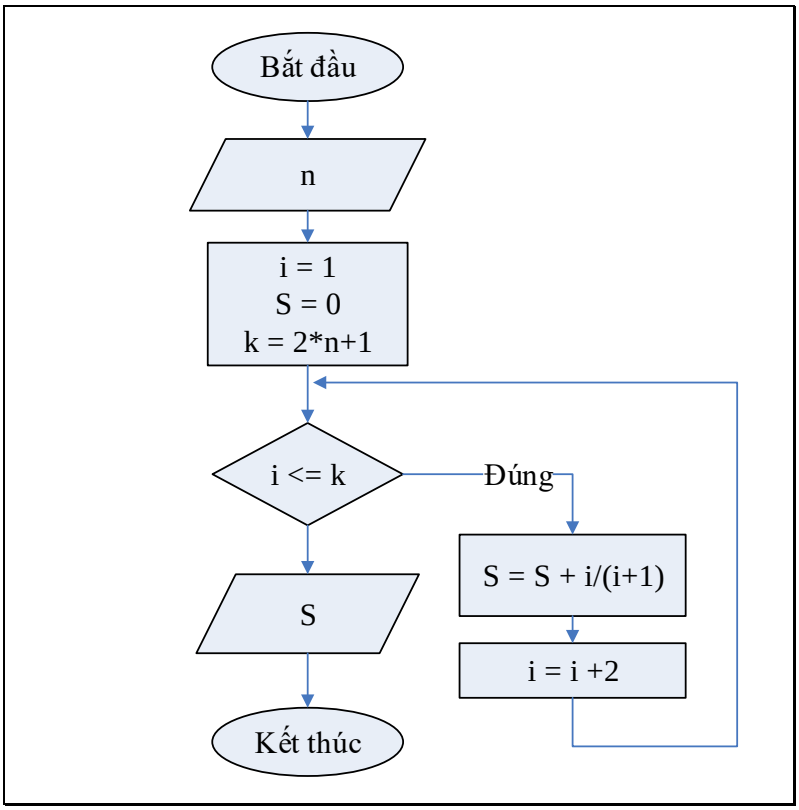
Ví dụ 2: Tính giá trị tuyệt đối của số nguyên x



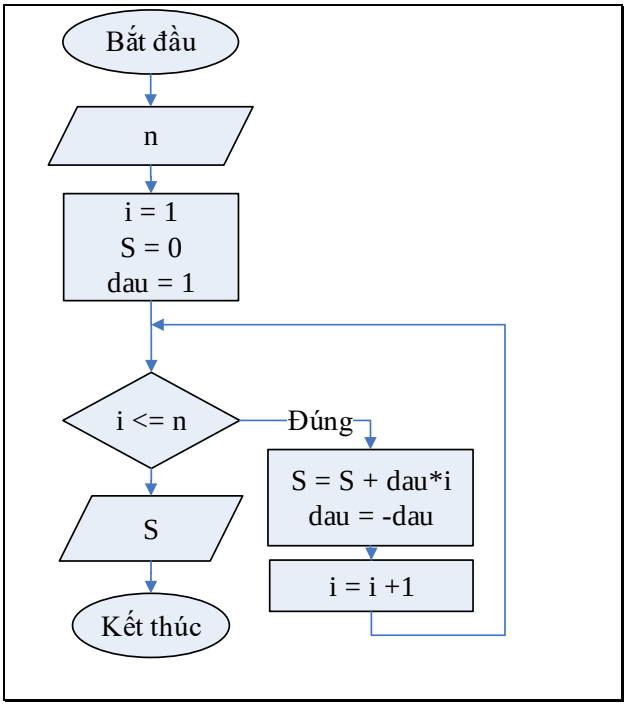
Ví dụ 3: Tính tổng: $S = 1 + 2 + 3 + \cdots + n$, với n>0



Ví dụ 4: Tính tổng: $S(n) = \frac{1}{2} + \frac{3}{4} + \frac{5}{6} + ... + \frac{2n+1}{2n+2}$, với n≥0



Ví dụ 5: Tính tổng: $S(n) = 1 - 2 + 3 - 4 + \cdots + (-1)^{n+1} n$, với n>0



II. Kiểu dữ liệu cơ bản

1) **Kiểu số nguyên:** Kiểu số nguyên là kiểu dữ liệu dùng để lưu các giá trị nguyên hay còn gọi là các giá trị rời rạc.

Stt	Tên kiểu	Ý nghĩa	Kích thước	Miền giá trị
1	char	Ký tự	1 byte	Từ -128 đến 127
		Số nguyên 1 byte	1 byte	
2	unsigned char	Số nguyên dương	1 byte	Từ 0 đến 255 (*tương đương 256 ký tự trong bảng mã ASCII*)
3	int	Số nguyên	2 bytes	Từ -32,768 đến 32,767
4	unsigned int	Số nguyên dương	2 bytes	Từ 0 đến 65,535
5	long	Số nguyên	4 bytes	Từ -2,147,483,648 đến 2,147,483,647
6	unsigned long	Số nguyên dương	4 bytes	Từ 0 đến 4,294,967,295

2) **Kiểu số thực**

Stt	Tên kiểu	Ý nghĩa	Kích thước	Miền giá trị
1	float	Số thực độ chính xác đơn	4 bytes	Từ 3.4×10^{-38} đến 3.4×10^{38}
2	double	Số thực độ chính xác kép	8 bytes	Từ 1.7×10^{-308} đến 1.7×10^{308}
3	long double	Số thực độ chính xác kép	10 bytes	Từ 3.4×10^{-4932} đến 1.1×10^{4932}

III. Biến và khai báo biến

Ví dụ:

- Khai báo biến a để lưu số nguyên 2 bytes: *int ia;*
- Khai báo biến tb để lưu số thực 4 bytes: *float ftb;*
- Khai báo các biến a, b, c cùng kiểu số nguyên 2 bytes: *int ia, ib, ic;*
- Khai báo các biến x, y, z có cùng kiểu số thực 4 bytes: *float fx, fy, fz;*
- Gán giá trị ban đầu cho biến: *int ia = 5; float fb = 3.4, c = 9.2; char cc = 'n';*
- Ép **fb** về phần nguyên nên **ia** sẽ có giá trị bằng 3: *int ia = (int)fk;*
- Định nghĩa hằng số MAX có giá trị là 100: *#define MAX 100* hoặc *const int MAX = 100;*

IV. Ký hiệu các phép toán

1) Phép toán số học

Stt	Ký hiệu	Ý nghĩa	Ghi chú
1	+	Phép cộng	
2	-	Phép trừ	
3	*	Phép nhân	
4	/	Phép chia	Đối với 2 số nguyên thì kết quả là chia lấy phần nguyên
5	%	Phép chia lấy phần dư	Chỉ áp dụng cho 2 số nguyên
6	++	Tăng một đơn vị nguyên	Nếu toán tử tăng giảm đặt trước thì tăng giảm trước rồi tính biểu thức hoặc ngược lại.
7	--	Giảm một đơn vị nguyên	

Một số ví dụ:

Ví dụ	Ý nghĩa
int ix = 5/2;	ix = 2, do 5 và 2 đều là số nguyên nên phép chia / được xem là phép chia lấy phần nguyên
int ix = 5%2;	ix = 2, do phép chia % là phép chia lấy phần dư
float fy = 5/2;	fy = 2.0, do 5 và 2 đều là số nguyên nên phép chia / được xem là phép chia lấy phần nguyên
float fy = 5/2.0	fy = 2.5, do 2.0 là hằng số thực nên phép chia / được xem là phép chia bình thường
int ia = 5; ia++;	ia = ia + 1 = 6
int ia = 5; *++ia;*	ia = ia + 1 = 6, do chỉ có một biến ia nên ký hiệu ++ nằm phía trước hay phía sau đều như nhau

int ia = 5; *int ib = ia++ + 7;*	- Tính ib = ia + 7 = 5 + 7 = 12 trước, do ký hiệu ++ nằm sau ia - Sau đó tính ia ++ = ia + 1 = 5 + 1 = 6
int ix = 5, iy = 11; *int iz = --ix + iy++;*	- Tính ix = ix - 1 = 4 trước, do ký hiệu -- nằm trước ix - Sau đó tính iz = ix + iy = 4 + 11 = 15 - Cuối cùng tính iy = iy + 1 = 11 + 1 = 12, do dấu ++ nằm sau iy

2) Phép toán so sánh và kết hợp so sánh

Stt	Ký hiệu	Ý nghĩa
1	>	Lớn hơn
2	<	Nhỏ hơn
3	>=	Lớn hơn hoặc bằng
4	<=	Nhỏ hơn hoặc bằng
5	= =	Bằng nhau
6	!=	Khác nhau
7	!	Phủ định phép so sánh
8	&&	Kết hợp theo nguyên tắc AND các phép so sánh
9	||	Kết hợp theo nguyên tắc OR các phép so sánh

3) Toán tử điều kiện

(Biểu thức điều kiện)? <Biểu thức 1>: <Biểu thức 2>;

Ý nghĩa:

- Nếu biểu thức điều kiện cho kết quả đúng thì sẽ thực hiện Biểu thức 1.
- Ngược lại thì sẽ thực hiện Biểu thức 2.

Ví dụ:

```
int n;
(n%2==0)? n ++ : n --;
```

→ *nếu n = 10 thì giá trị n = 11*

→ *nếu n = 21 thì giá trị n = 20*

> ✍ *Biểu thức biểu kiện phải dùng ký hiệu so sánh, nhất là ký hiệu so sánh phép bằng nhau (==), sinh viên thường hay sai do viết chỉ có 1 dấu bằng* ***(dấu = là phép gán của C)***

4) Thứ tự ưu tiên các phép toán

Trong biểu thức có kết hợp nhiều phép toán thì ngôn ngữ C sẽ dựa vào bảng thứ tự ưu tiên để xác định thứ tự thực hiện.

Toán tử	Độ ưu tiên	Trình tự kết hợp
() [] ->	1	Từ trái qua phải
! ~ ++ -- - + * & sizeof	2	Từ phải qua trái
* / %	3	Từ trái qua phải
+ -	4	Từ trái qua phải
<< >>	5	Từ trái qua phải
< <= >= >	6	Từ trái qua phải
== !=	7	Từ trái qua phải
&	8	Từ trái qua phải
|	9	Từ trái qua phải
^	10	Từ trái qua phải
&&	11	Từ trái qua phải
||	12	Từ trái qua phải
? :	13	Từ phải qua trái
= += -= *= /= %=	14	Từ phải qua trái

5) Chuỗi định dạng (nếu dùng *printf*)

Stt	Kiểu	Chuỗi định dạng	5	long	%ld
1	char	%c	6	unsigned long	%lu
		%d	7	char *	%s
2	unsigned char	%d	8	float	%f
3	int	%d	9	double	%lf
4	unsigned int	%u	10	long double	%lf

V. Các hàm cơ bản khác: Giới thiệu một số hàm cơ bản cần sử dụng trong một số bài tập của giáo trình. Các hàm này đều nằm trong thư viện <math.h>

Stt	Tên hàm	Ý nghĩa	Ví dụ	Kết quả
1	abs	Tính trị tuyệt đối của một số	*int iy = abs(-7);*	iy = |-7| = 7
2	pow	Tính mũ	*int iy = pow(2, 3);*	iy = 2^3 = 8
3	sqrt	Tính căn bậc hai của x	*float fcan = sqrt(4);*	fcan = $\sqrt{4}$ = 2

VI. Cấu trúc rẽ nhánh

1) Cấu trúc if…else

```
if (biểu thức điều kiện)
{
    <lệnh hoặc khối lệnh 1>;
}
else
{
    <lệnh hoặc khối lệnh 2>;
}
```

Ví dụ: *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");
   }
}
```

2) **Cấu trúc if … else lồng nhau:** Nếu cần xét nhiều trường thì có thể sử dụng cấu trúc if…else lồng nhau.
   Ví dụ: *Giải và biện luận phương trình bậc nhất: ax+b=0*

```
void main ()
{
   float a, b;
   printf ( "\n Nhap vao a:");
   scanf ( "%f", &a);
   printf ( " Nhap vao b:");
   scanf ( "%f", &b) ;
   if (a==0)
   {    if (b==0)
        {
             printf ( " \n PTVSN");
        }
        else
        {
             printf ( " \n PTVN");
        }
    }
      else
   {    printf (" \n Nghiem x=%f", -b/a);
   }
   _getch ();
}
```

VII. Cấu trúc lựa chọn switch…case: Sử dụng cấu trúc này khi cần thực hiện một khối lệnh của một trường hợp trong nhiều trường hợp, mỗi trường hợp tương ứng với một giá trị nguyên hay một ký tự cụ thể.

```
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]
}
```

- n_i là các **hằng số nguyên hoặc ký tự**.

Ví dụ: Nhập vào một số nguyên tương ứng với các món ăn, in ra tên của món đã chọn.

```
void main()
{    int ichon ;
     printf ("***THUC DON***") ;
     printf ("\n1. Lau thai!");
     printf ("\n2. Nuoc ngot!");
     printf ("\n3. Ca loc hap bau!");
     printf ("\n ==>> Xin moi ban chon mon an: ");
     scanf ("%d",&ichon);
     switch (ichon)
     {    case 1:
               printf ("\nBan chon lau thai!");
               break;
          case 2:
               printf ("\nBan chon nuoc ngot!");
               break;
          case 3:
               printf ("\nBan chon ca loc hap bau!");
               break;
          default:
               printf ("\nBan chon khong dung!");
     }
     getch();
}
```

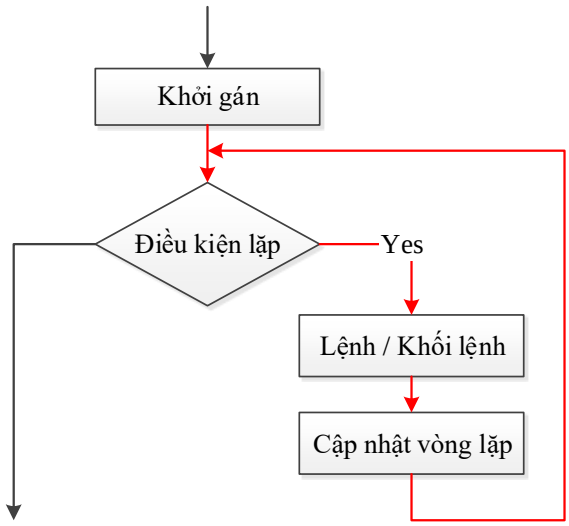
VIII. Cấu trúc lặp: dùng để thực hiện một lệnh hay khối lệnh nhiều lần tùy thuộc vào điều kiện của vòng lặp. Ngôn ngữ C có hai dạng cấu trúc lặp: lặp kiểm tra điều kiện trước khi lặp (for, while) và lặp kiểm tra điều kiện sau (do…while).

1) Cấu trúc lặp for

```
for (<Khởi gán>; <Điều kiện lặp>; <Cập nhật vòng lặp>)
{
    <khối lệnh>;
}
```

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

Hoạt động của cấu trúc điều khiển for:



Bước 1: *Khởi gán*
Bước 2: *Kiểm tra điều kiện lặp*

- *Nếu **đúng** thì cho thực hiện các lệnh của vòng lặp, thực hiện cập nhật vòng lặp. Quay trở lại bước 2.*
- *Ngược lại thoát khỏi lặp.*

Ví dụ: *In ra màn hình bảng mã ASCII từ ký tự số 33 đến 255.*

```
void main()
{
   for (int i=33;i<=255;i++)
   {
        printf("Ma ASCII cua %c: %d\t", i, i) ;
   }
      _getch() ;
}
```

2) Cấu trúc lặp while

```
< Khởi gán>;
while (<Điều kiện lặp>)
{
    lệnh/ khối lệnh;
    <Cập nhật vòng lặp>;
}
```

> ✍ ***Lưu ý***: *Cách hoạt động của* **while** *giống như cấu trúc* **for**

Ví dụ: *Tính giá trị trung bình các chữ số của số nguyên n gồm k chữ số.*

```
void main()
{
   long n, tong=0;
   int sochuso=0;
   float tb;
   printf("Nhap vao gia tri n gom k chu so");
   scanf("%ld",&n);
   while(n>0)
   {
        tong=tong+n%10;
        sochuso++;
        n=n/10;
   }
   tb=1.0*tong/sochuso;
   printf ("Gia tri trung binh la: %f", tb);
   _getch ();
}
```

3) Cấu trúc lặp do…while

```
<Khởi gán>;
do
{
    < lệnh hoặc khối lệnh>;
   <Cập nhật vòng lặp>;
} while (<Điều kiện lặp>);
```

Thực hiện khối lệnh bên trong vòng lặp cho đến khi gặp điều kiện sai thì dừng.

Ví dụ: *Nhập ký tự từ bàn phím hiển thị lên màn hình mã ASCII của ký tự đó, thực hiện đến khi nhấn phím ESC (Mã ASCII của phím ESC là 27).*

```
void main()
{
   int ma ;
   do{
        ma=_getch();
        if (ma !=27)
        {
             printf ("Ma ASCII %c:%d\t", ma, ma);
        }
   } while(ma!=27);
}
```

✍ *Lặp* ***while*** *kiểm tra điều kiện trước khi thực hiện lặp, còn 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.*

IX. Lệnh break và continue

1) **Lệnh break:** Dùng để kết thúc vòng lặp trực tiếp chứa nó khi thỏa một điều kiện nào đó.

Ví dụ: *Cho phép người dùng nhập liên tục giá trị n cho đến khi nhập âm thì dừng.*

```
void main
{
 while (1)
 {
      printf("\nNhap n: ");
      scanf("%d", &n);
      if(n<0)
      {
           break;
      }
 }
}
```

2) **Lệnh continue:** Dùng để bỏ qua một lần lặp khi thỏa điều kiện nào đó.

Ví dụ: *In ra màn hình giá trị từ 10 đến 20 trừ đi số 13 và số 17.*

```
void main()
{ for(int i=10 ; i<=20; i++)
 {    if(i==13 || i==17)
      {
           continue;
      }
      printf("%d\t", i);
 }
}
```