

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.1 Lệnh đơn và lệnh phức

8.2 Lệnh IF

8.3 Lệnh SWITCH-CASE

8.4 Lệnh WHILE

8.5 Lệnh DO-WHILE

8.6 Lệnh FOR

8.7 Lệnh BREAK và lệnh
CONTINUE

8.8 Lệnh RETURN

8.9 Lệnh GOTO

8.10 Lệnh RỖNG

Bài tập cuối chương

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.1 LỆNH ĐƠN VÀ LỆNH PHỨC

(SIMPLE STATEMENT VÀ COMPOUND STATEMENT)

- Lệnh đơn là một biểu thức thuộc loại bất kỳ theo sau nó là một dấu chấm phẩy (;), do đó lệnh đơn còn được gọi là lệnh biểu thức.

Ví dụ: Các lệnh sau đây là các lệnh đơn

```
a = a + 1;
```

```
b >>= 3;
```

```
printf (...);
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.1 LỆNH ĐƠN VÀ LỆNH PHỨC

(SIMPLE STATEMENT VÀ COMPOUND STATEMENT)

-Lệnh phức bao hàm một hay nhiều lệnh đơn được bao bên trong cặp dấu ngoặc nhọn ({ }) và được bộ dịch C xem như là một lệnh đơn.

Ví dụ: Xét lệnh if sau

```
if (a > 0)
```

```
{
```

```
    i += 2;
```

```
    a++;
```

```
    n = a * i;
```

```
}
```

→ lệnh phức, được xem là một lệnh

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.1 LỆNH ĐƠN VÀ LỆNH PHỨC

(SIMPLE STATEMENT VÀ COMPOUND STATEMENT)

Các lệnh điều khiển này có thể được chia ra làm hai nhóm:

- Nhóm lệnh liên quan đến việc rẽ nhánh chương trình: if-else, switch-case, goto,...
- Nhóm lệnh lặp: while, for, do_while

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.2 LỆNH IF

Lệnh **if** cho phép lập trình viên thực hiện một lệnh đơn hay một lệnh phức tùy theo biểu thức điều kiện, nếu **biểu thức có trị khác 0** thì lệnh được thực thi.

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

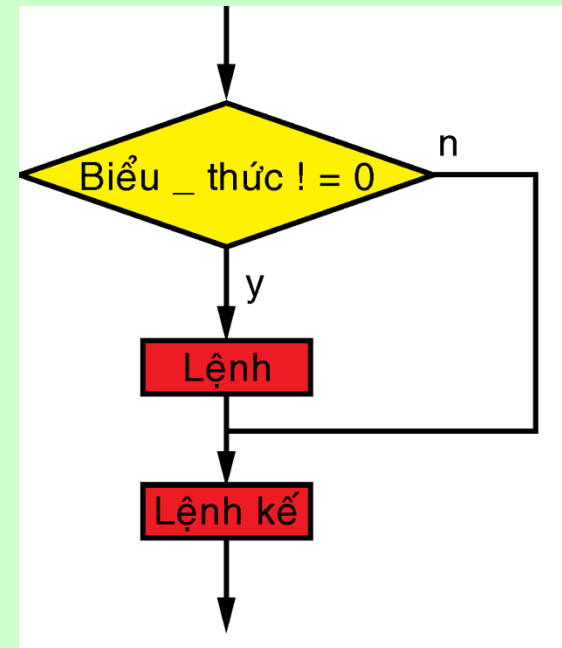
8.2 LỆNH IF

Dạng 1:

if (biểu_thức)

lệnh;

- **biểu_thức** là một biểu thức bất kỳ, có thể có hằng, biến hoặc gọi hàm trong đó và sau cùng là biểu thức này sẽ có trị 0 hoặc 1



CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

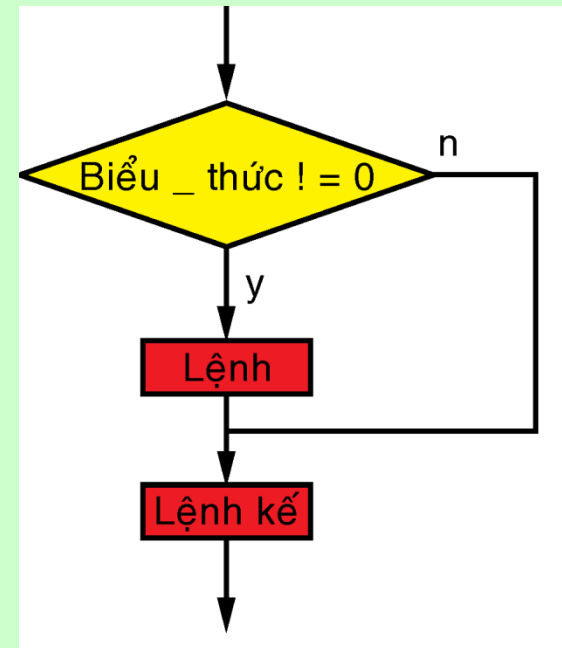
8.2 LỆNH IF

Dạng 1:

if (biểu_thức)

lệnh;

- **lệnh** là lệnh thực thi của if, có thể là lệnh đơn, phức hoặc lệnh rỗng.



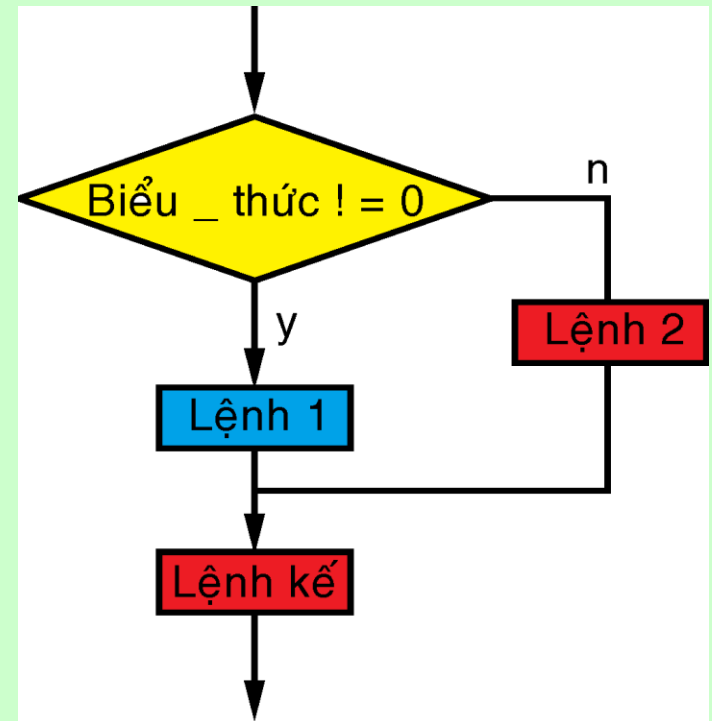
CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.2 LỆNH IF

Dạng 2:

```
if (biểu_thức)  
    lệnh_1;  
else  
    lệnh_2;
```



CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.2 LỆNH IF

Ví dụ: Xét chương trình sau đây

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

```
#include <conio.h>
```

```
main()
```

```
{    int n;
    clrscr();
    printf (Moi nhap mot so: );
    scanf (%d, &n);
    if (n % 2 == 0)
        printf ("So la so chan \n");
    printf ("Moi ban nhan mot phim de ket thuc \n");
        getch();    }
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.2 LỆNH IF

Ví dụ: Xét chương trình sau đây

```
#include <stdio.h>
#include <conio.h>
main()
{
    int n; clrscr();
    printf ("Moi nhap mot so: "); scanf ("%d", &n);
    if (n % 2 == 0)
        printf ("So la so chan \n");    ←vẫn có dấu chấm
phẩy
    else
        printf ("So la so le \n");
    printf ("Moi ban nhan mot phim de ket thuc \n");
    getch();
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.2 LỆNH IF

Ví dụ: Xét chương trình sau đây

```
if (a > 0)
  if (b > 0)
    c = b + a;
  else
    c = b - a;
```

```
if (a > 0)
{
  if (b > 0)
    c = b - a;
}
else
  c = b - a;
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.2 LỆNH IF

```
if (biểu_thức_1)
    lenh_1;
else if (biểu_thức_2)
    lenh_2;
else if (biểu_thức_3)
    lenh_3;
.....
else
    lenh_n;
```

Khi thực hiện lệnh if_else lồng nhau như thế này các biểu thức sẽ được tính lần lượt từ trên xuống dưới nếu có biểu thức nào khác 0, lệnh tương ứng với if đó sẽ được thi hành và toàn bộ phần còn lại của lệnh if-else được bỏ qua.

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.2 LỆNH IF

Ví dụ: Chương trình ví dụ sau nhập vào một ký tự, kiểm tra ký tự đó là thường, hoa, ký số hoặc ký tự kết thúc file hay ký tự khác.

```
#include <stdio.h>
#include <conio.h>
main()
    {
        char c;
        clrscr();
        printf ("Nhap mot ky tu: ");
        c = getch();
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.2 LỆNH IF

```
if (c == EOF)
    printf ("Da den cuoi file \n");
else if (c >= 'a' && c <= 'z')
    printf ("ky tu thuong\n");
else if (c >= 'A' && c <= 'Z')
    printf ("ky tu hoa\n");
else if (c >= '0' && c <= '9')
    printf ("ky tu so\n");
else
    printf ("ky tu khac\n");
getch();
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.3 LỆNH SWITCH-CASE

switch (biểu_thức)

{

case hằng_1:

lệnh_1;

break;

case hằng_2:

lệnh_2;

break;

:

:

case hằng_n:

lệnh_n;

break;

default:

lệnh;

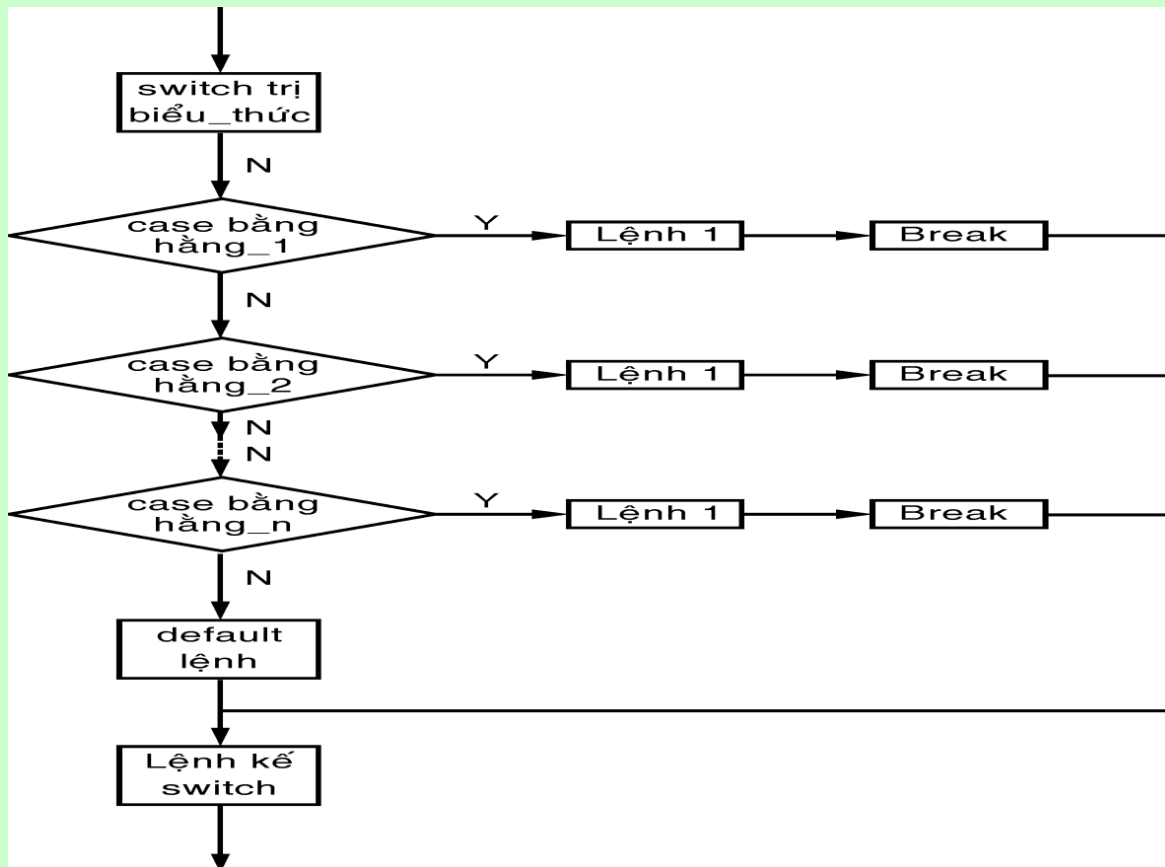
break;

}

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.3 LỆNH SWITCH-CASE



CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.3 LỆNH SWITCH-CASE

Ví dụ: Viết chương trình nhập một trị, nếu trị đó chia hết cho 5 thì cộng thêm 5 vào cho số đó, nếu trị đó chia cho 5 dư 1 thì cộng thêm 1, tương tự cho 3, nếu là số khác thì báo không thỏa.

```
#include <stdio.h>
#include <conio.h>
main()
```

```
{    int so;
      clrscr();
      printf ("Nhap mot so: ");
      scanf ("%d", &so);
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.3 LỆNH SWITCH-CASE

```
switch (so % 5)
{
    case 0:
        so += 5;
        printf ("Tri la: %d\n", so);
        break;
    case 1:
        so += 1;
        printf ("Tri la: %d\n", so);
        break;
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.3 LỆNH SWITCH-CASE

```
        case 3:
            so += 3;
            printf ("Tri la: %d\n", so);
            break;
        default:
            printf ("Khong thoa\n");
            break;
    }
    getch();
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.3 LỆNH SWITCH-CASE

Lệnh **break** cuối mỗi case sẽ chuyển điều khiển chương trình ra khỏi lệnh switch. Nếu không có break, các lệnh tiếp ngay sau sẽ được thực thi dù các lệnh này có thể là của một case khác.

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.3 LỆNH SWITCH-CASE

Ví dụ: Xét ví dụ nhập tháng và năm, kiểm tra số ngày trong tháng.

```
switch (thang)
```

```
{  
    case 4:
```

```
    case 6:
```

```
    case 9:
```

```
    case 11:
```

```
        so_ngay = 30;
```

```
        break;
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.3 LỆNH SWITCH-CASE

case 2:

```
if (nam % 4 == 0)
    so_ngay = 29;
else
    so_ngay = 28;
break;
```

default:

```
so_ngay = 31;
break;    }
```

```
printf("Thang %d nam %d co %d ngay\n", thang, nam,
so_ngay);
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.4 LỆNH WHILE

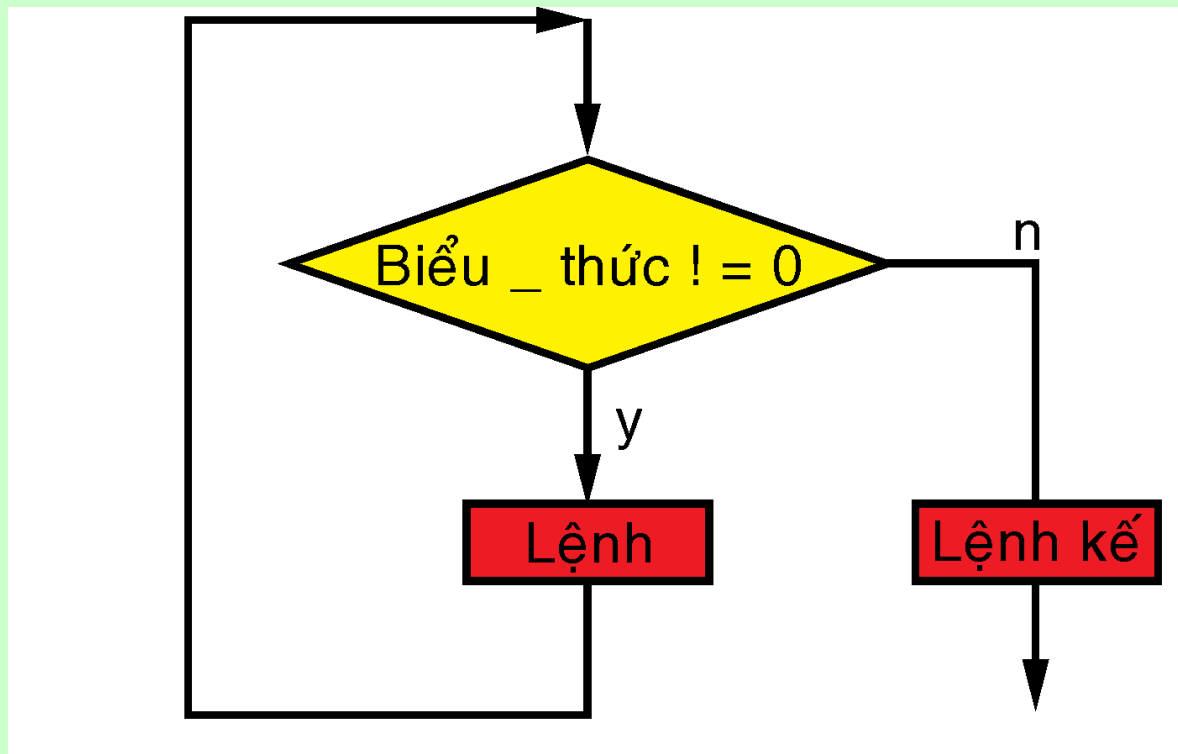
Có thể nói while là lệnh lặp cơ bản của ngôn ngữ lập trình có cấu trúc, nó cho phép chúng ta lặp lại một lệnh hay một nhóm lệnh **trong khi** điều kiện còn đúng (true-tức khác 0). Cú pháp của lệnh while:

while (biểu-thức) lệnh

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.4 LỆNH WHILE



CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.4 LỆNH WHILE

Ví dụ: Chương trình sau đây sẽ in ra màn hình 10 số ngẫu nhiên từ 0 đến 99.

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>>
#include <time.h>
main()
    {
        int i = 1;
        clrscr();
        randomize();
        printf ("Số ngẫu nhiên trong khoảng 0-
99 là: ");
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.4 LỆNH WHILE

```
while (i <= 10)
{
    printf ("%d", random(100));
    i++;
}
getch();
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.4 LỆNH WHILE

Cách khác:

```
int i = 10;
clrscr();
randomize();
printf ("Số ngẫu nhiên trong khoảng 0-99 là: ");
while (i)
    {
        printf ("%d", random(100));
        --i;
    }
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.4 LỆNH WHILE

Ví dụ: Nhập các ký tự cho đến khi nào nhận được ký tự ESC có mã ASCII là 27 thì kết thúc chương trình.

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

```
#include <conio.h>
```

```
#define ESC 27
```

```
main()
```

```
{
```

```
    char c;
```

```
    clrscr();
```

```
    printf ("Cac ky tu duoc nhap la: ");
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.4 LỆNH WHILE

```
while (1)
{
    c = getch();
    if (c == ESC)
        break;
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.4 LỆNH WHILE

```
#include <stdio.h>
#include <conio.h>
#define ESC 27
main()
{
    char c;
    clrscr();
    printf ("Cac ky tu duoc nhap la: ");
    while (getche() - ESC)
        ;           ←lệnh thực thi rỗng
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.5 LỆNH DO-WHILE

Nếu lệnh **while** cho phép kiểm tra điều kiện trước rồi thực thi lệnh sau, như vậy ngay từ đầu mà điều kiện đã sai thì lệnh của **while** không được thực thi, thì lệnh lặp **do-while** lại thực thi lệnh trước rồi mới kiểm tra điều kiện sau.

Cú pháp của lệnh **do-while** như sau:

do

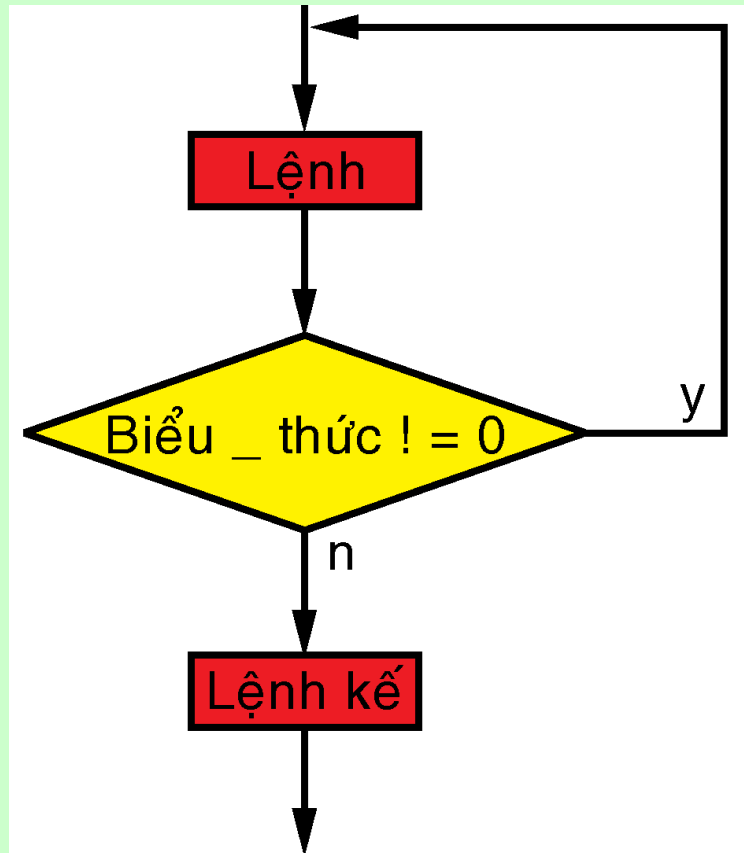
lệnh

while (biểu_thức);

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.5 LỆNH DO-WHILE



CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.5 LỆNH DO-WHILE

Ví dụ: Viết chương trình cho phép kiểm tra và in ra phím mũi tên đã được nhấn.

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

```
#include <conio.h>
```

```
#define ESC 27
```

```
main()
```

```
{
```

```
    char c;
```

```
    clrscr();
```

```
    printf ("\n Moi an cac phim mui ten \n");
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.5 LỆNH DO-WHILE

```
do
{
    c = getch();
    if (c == 0)
    {
        c = getch();
        switch(c)
        {
            case 'H':
                printf ("Ban da an mui ten
len\n");
                break;
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.5 LỆNH DO-WHILE

```
case 'P':  
    printf ("Ban da an mui ten  
xuong\n");  
  
    break;  
case 'K':  
    printf ("Ban da an mui ten qua  
trai\n");  
  
    break;  
case 'M':  
    printf ("Ban da an mui ten qua  
phai\n");
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.5 LỆNH DO-WHILE

```
break;
```

```
        } /* end switch */
```

```
    }
```

```
    }while (c != 27);
```

```
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.5 LỆNH DO-WHILE

Chú ý rằng mỗi phím mũi tên khi được ấn đều sinh ra hai ký tự: ký tự đầu luôn là ký tự có mã ASCII là 0 (tức ký tự NUL), ký tự thứ hai là các mã ASCII tương ứng với phím, trong ví dụ trên thì

- + Phím mũi tên lên có mã là 0 và 'H'
- + Phím mũi tên xuống có mã là 0 và 'P'
- + Phím mũi tên qua trái có mã là 0 và 'K'
- + Phím mũi tên có mã là 0 và 'M'.

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.6 LỆNH FOR

Tương tự như ngôn ngữ PASCAL, trong ngôn ngữ C cũng có vòng lặp **for**, đây cũng là một lệnh lặp cho phép kiểm tra điều kiện trước, giống như **while**. Cú pháp của lệnh **for** như sau:

```
for (biểu_thức1; biểu_thức2; biểu_thức3)  
    lệnh
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

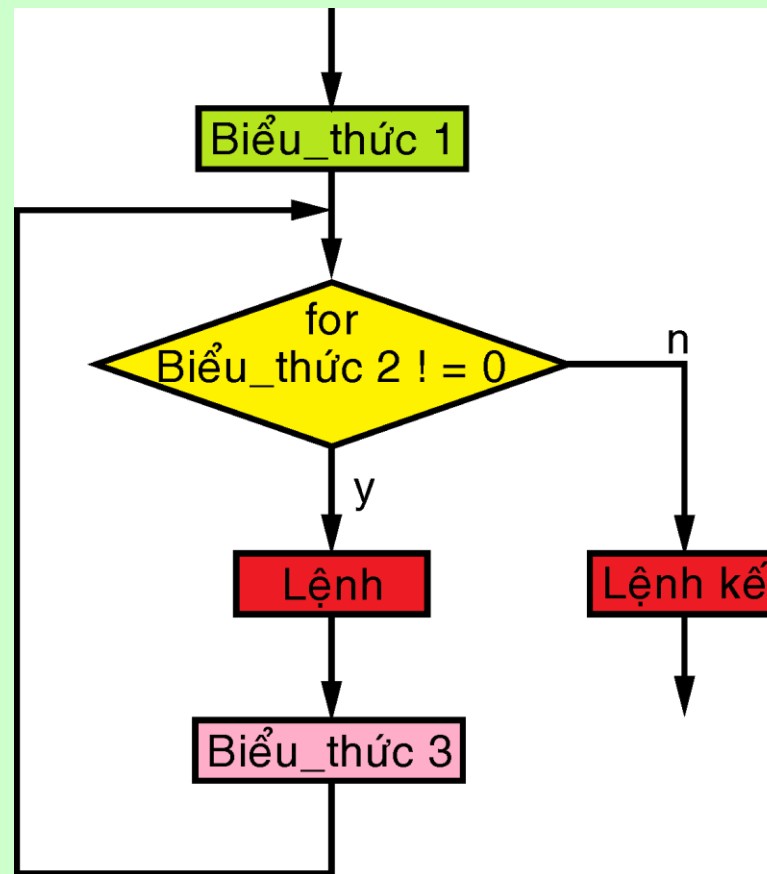
8.6 LỆNH FOR

- **biểu_thức1** có ý nghĩa là biểu thức để khởi động trị đầu cho biến điều khiển vòng for, nó có thể là biểu thức gán hay biểu thức phẩy, có thể không có.
- **biểu_thức2** có ý nghĩa là biểu thức cho phép kiểm tra xem vòng lặp có được tiếp tục lặp nữa hay không.
- **biểu_thức3** là biểu thức có ý nghĩa cho phép thay đổi biến điều khiển vòng lặp để vòng lặp tiến dần đến kết thúc. Biểu thức này được tính sau khi các lệnh thực thi trong thân vòng for được thực hiện xong.

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.6 LỆNH FOR



CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.6 LỆNH FOR

Ví dụ: vòng lặp for để tính tổng từ 1 tới n như sau

```
s = 0;
```

```
for (i = 1; i <= n; i++)
```

```
    s += i;
```

Có thể viết ngắn gọn hơn như sau

```
for (i = 1, s = 0; i <= n; i++)
```

```
    s += i;
```

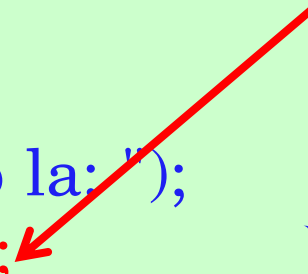
CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.6 LỆNH FOR

Ví dụ: Nhập các ký tự cho đến khi nào nhận được ký tự ESC có mã ASCII là 27 thì kết thúc chương trình.

```
#include <stdio.h>
#include <conio.h>
#define ESC 27
main()
{
    char c;
    clrscr();
    printf ("Cac ky tu duoc nhap la: ");
    for ( ; (c = getch()) != ESC;) ; }
```



CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.7 LỆNH BREAK VÀ LỆNH CONTINUE

Đây là hai lệnh nhảy không điều kiện của C, chúng cho phép lập trình viên có thể thay đổi tiến trình lặp của các cấu trúc lặp mà ta đã biết: for, while, do-while.

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.7 LỆNH BREAK VÀ LỆNH CONTINUE

1. Lệnh break

Trong cấu trúc `switch-case`, lệnh `break` sẽ kết thúc lệnh `switch-case`; còn trong các cấu trúc lặp thì lệnh `break` cho phép thoát sớm ra khỏi vòng lặp (`while`, `for` hoặc `do-while`) chứa nó mà không cần xét điều kiện của lệnh kế tiếp sau vòng lặp.

Cú pháp của lệnh `break`:

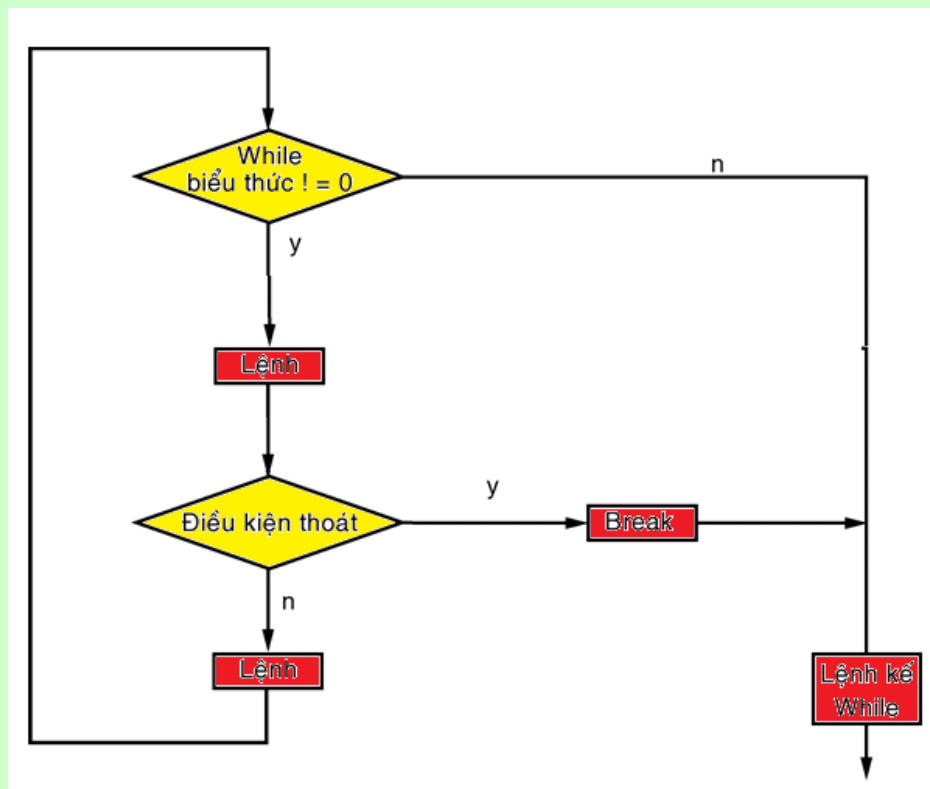
`break;`

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.7 LỆNH BREAK VÀ LỆNH CONTINUE

1. Lệnh break



CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.7 LỆNH BREAK VÀ LỆNH CONTINUE

1. Lệnh break

Ví dụ:

```
#include <stdio.h>
#include <conio.h>
#define TAB '\t'
#define ESC '\x1b'
#define ENTER '\r'
main()
{
    char c;
    clrscr();
    while (1)
    {
        printf ("Moi ban nhap day cac ky tu: ");
        switch (c = getche())
        {
            case TAB:
                printf ("Ban da an phim TAB \n");
                break; _____
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.7 LỆNH BREAK VÀ LỆNH CONTINUE

1. Lệnh break

Ví dụ:

```
        case ENTER:
            printf ("Ban da an phim ENTER \n");
            break;
        case ESC:
            printf ("Ban da an phim ESC \n");
            break;
        default:
            printf ("Ban da an phim khac \n");
            break;
    } /* kết thúc switch */
if ( c == ESC)
    break;
} /* kết thúc while */
} /* kết thúc chương trình */
```

break để thoát ra khỏi switch

→ break để thoát khỏi while

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.7 LỆNH BREAK VÀ LỆNH CONTINUE

2. Lệnh continue

lệnh **continue** có tác dụng chuyển điều khiển chương trình về đầu vòng lặp chuẩn bị cho chu kỳ lặp mới, bỏ qua các lệnh còn lại nằm ngay sau lệnh nó trong chu kỳ lặp hiện hành. Lệnh này chỉ được dùng trong các vòng lặp, để bỏ qua các lệnh không cần thực thi trong vòng lặp khi cần thiết.

Cú pháp lệnh continue:

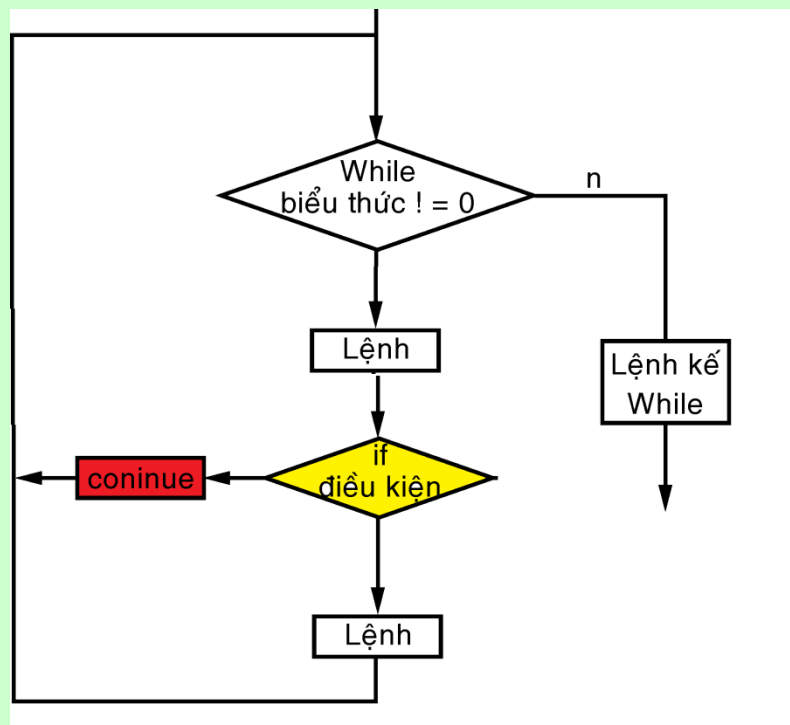
continue;

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.7 LỆNH BREAK VÀ LỆNH CONTINUE

2. Lệnh continue



CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.7 LỆNH BREAK VÀ LỆNH CONTINUE

2. Lệnh continue

Ví dụ:

```
i = 0;
while (i <= 10)
{
    i ++;
    if (i >= 6 && i <= 8)
        continue;
    printf ("Trị hiện thời của i là %d\n", i);
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.7 LỆNH BREAK VÀ LỆNH CONTINUE

2. Lệnh continue

Ví dụ: Viết chương trình nhập một dãy số, tính tổng của các số dương trong dãy số đó và thương số của tổng đó với từng số dương này.

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.7 LỆNH BREAK VÀ LỆNH CONTINUE

2. Lệnh continue

```
#include <stdio.h>
#include <conio.h>
main()
{
    double a[100];
    double tong;
    int i, n;
    clrscr();
    printf ("Co bao nhieu so can tinh: ");
    scanf ("%d", &n);
    printf ("Nhap cac so can tinh tong: ");
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.7 LỆNH BREAK VÀ LỆNH CONTINUE

2. Lệnh continue

```
for (i = 0; i < n; i++)
    scanf ("%lf", &a[i]);
for (i = 0, tong = 0; i < n; i++)
{
    if (a[i] <= 0)
        continue;
    tong += a[i];
}
printf ("Tong cua cac so duong la %.2lf\n",
tong);
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.7 LỆNH BREAK VÀ LỆNH CONTINUE

2. Lệnh continue

```
for (i = 0; i < n; i++)
{
    if (a[i] <= 0)
        continue;
    printf("Thuong cua tong voi so thu
%d la %5.2lf\n",i,tong/a[i]);
}
getch();
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

2.8 LỆNH RETURN

Lệnh này dùng để thoát ra khỏi hàm hiện thời trở về hàm đã gọi nó, có thể trả về cho hàm gọi một trị. Lệnh này sẽ kết thúc hàm dù nó nằm ở đâu trong thân hàm. Khi gặp lệnh này C sẽ không thực hiện bất cứ lệnh nào sau lệnh return nữa. Các cú pháp của lệnh return như sau:

return;

return (biểu-thức);

return biểu-thức;

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.8 LỆNH RETURN

Ví dụ:

Thiết kế hàm trả về kết quả so sánh hai số theo quy tắc sau đây:

số đầu > số sau: hàm trả về trị 1

số đầu = số sau: hàm trả về trị 0

số đầu < số sau: hàm trả về trị -1

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.8 LỆNH RETURN

Ví dụ:

```
int so_sanh (int a, int b)
{
    if (a > b)/* Lệnh return kết thúc hàm, trả về trị i
cho */
        return 1;    /* nơi đã gọi hàm */
    else if (a == b)
        return 0; /* Trả về trị 0 cho nơi gọi hàm khi a
= b */
    else /* a < b */
        return -1; /* Trả về trị -1 cho nơi gọi hàm khi
a < b */
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.8 LỆNH RETURN

Ví dụ:

```
int so_sanh (int a, int b)
{
    if (a > b)    /* Lệnh return kết thúc hàm, trả về trị
1 cho */
        return 1;    /* nơi đã gọi hàm */
    else if (a == b)
        return 0; /* Trả về trị 0 cho nơi gọi hàm khi
a = b */
    return -1;    /* Trả về trị -1 cho nơi gọi hàm khi a
< b */
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.8 LỆNH RETURN

Ví dụ:

```
int so_sanh (int a, int b)
{
    return (a > b) ? 1 : (a == b) ? 0 : -1;
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.8 LỆNH RETURN

Ví dụ:

Chương trình sau dùng lệnh return để kết thúc vòng lặp lặp vô tận khi điều kiện thỏa (là phím ESC được nhấn).

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.8 LỆNH RETURN

Ví dụ:

```
#include <stdio.h>
#include <conio.h>
#define ESC '\x1b'
void nhan_ky_tu (void);           ← prototype của hàm
main()
{
    char c;
    clrscr();
    printf ("Moi ban nhap cac ky tu: ");
    nhan_ky_tu ();               ← gọi hàm
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.8 LỆNH RETURN

Ví dụ:

```
void nhan_ky_tu (void)           ←định nghĩa hàm
{
    while (1)
        if (getche() == ESC)
            return;
}
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.9 LỆNH GOTO

Mặc dù không ủng hộ cho việc lập trình có goto nhưng C vẫn có lệnh rẽ nhánh không điều kiện goto, lệnh này cho phép chuyển điều khiển chương trình cho một lệnh nào đó.

Cú pháp của lệnh **goto**:

goto nhãn;

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.9 LỆNH GOTO

Với nhãn là một danh hiệu không chuẩn, danh hiệu này sẽ được đặt ở trước lệnh mà ta muốn nhảy đến theo cú pháp sau:

nhãn: lệnh

nhãn mà lệnh **goto** muốn nhảy đến phải nằm trong cùng một hàm với lệnh **goto** đó, do đó trong những hàm khác nhau có thể có các tên nhãn giống nhau, nhưng trong cùng một hàm các tên nhãn này phải khác nhau.

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.9 LỆNH GOTO

Ví dụ:

Cách sử dụng lệnh goto trong một chương trình C
main()

```
lap_lai:      {  
              clrscr();  
              if ((c = getch()) != ESC) ...  
                goto lap_lai;  
            }
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.10 LỆNH RỖNG

Trong C có khái niệm **lệnh rỗng**, lệnh này chỉ có một dấu chấm phẩy (;), nó rất cần thiết trong nhiều trường hợp, như đối với các vòng lặp, khi ta đặt các lệnh biểu thức thực thi vào trong các biểu thức của lệnh thì ta không cần có thêm lệnh thực thi làm thân cho chúng nữa, **khi đó nếu để trống**, C sẽ hiểu nhầm rằng lệnh kế tiếp sẽ là thân của vòng lặp, do đó chỉ còn cách cho một lệnh rỗng làm thân của chúng.

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.10 LỆNH RỖNG

Ví dụ:

Vòng lặp for để tính giai thừa từ 1 tới n như sau

```
for (i = gt = 1; i <= n; gt *= i++)  
    ;  
printf ("Giai thua %d! = %d\n", n, gt);
```

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

8.10 LỆNH RỖNG

Ví dụ:

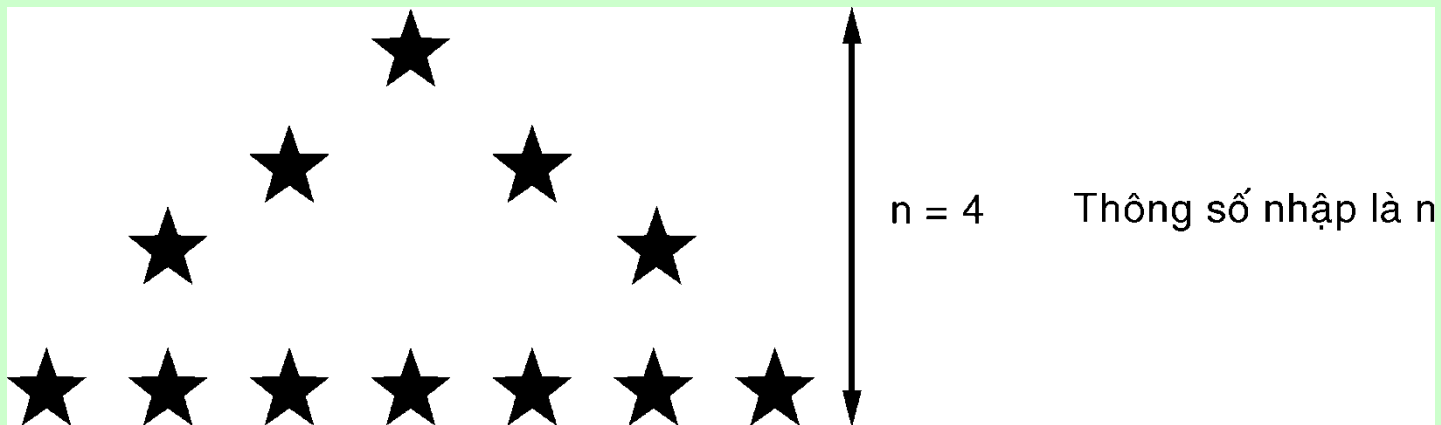
```
for (i =1,s = 0; i < 10; i++) ;    s += i;  
printf("Tong la %d \n",s);
```


CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

BÀI TẬP CUỐI CHƯƠNG

4. In ra màn hình bản cử chương cần biết.
5. In ra màn hình các bản cử chương từ 2 đến 9.
6. Vẽ ra màn hình hình sau:



CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

BÀI TẬP CUỐI CHƯƠNG

7. Tính biểu thức sau đây

a) $T = 1! + 2! + \dots + n!$ thông số nhập là n

b) $T = \frac{1! + (1+2)! + \dots + (1+\dots+n)!}{n!}$ thông số nhập là n

c) $T = \frac{e^1}{1!} + \frac{e^2}{2!} + \dots + \frac{e^n}{n!}$ thông số nhập là n

Biết trong C có hàm $\exp(x)$ để tính , prototype hàm này nằm trong file `math.h`.

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

BÀI TẬP CUỐI CHƯƠNG

8. Tính biểu thức sau:

$$s = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \dots + \frac{1}{n} < 2 \quad (1)$$

Hãy viết chương trình nhập một số a thỏa:

$1 < a < 2$, sau đó tìm số n thỏa điều kiện (1):

$$s < a$$

CHƯƠNG 8

CÁC LỆNH ĐIỀU KHIỂN VÀ VÒNG LẶP

BÀI TẬP CUỐI CHƯƠNG

9. Một người muốn gửi một số tiền vào ngân hàng, hãy viết chương trình tính tổng số tiền mà người đó có được sau khi đã gửi ngân hàng theo một trong hai cách gửi:

- Gửi từng tháng rút tiền lãi
- Gửi không rút lãi từng tháng, mà nhập lãi vào vốn

Thông số nhập cần thiết:- Số tiền gửi lúc đầu

- Thời gian gửi (theo tháng)
- Lãi suất/tháng