Chương 01.5: Các lệnh điều khiển chương trình

I. Lệnh lựa chọn

II. Lệnh lặp

III. Lệnh break

IV. Lệnh continue

I. Lệnh lựa chọn

1. Lệnh kiểm tra điều kiện if

2. Lệnh thử và rẽ nhánh switch

1. Lệnh kiểm tra điều kiện if

- Lệnh này có 2 dạng:

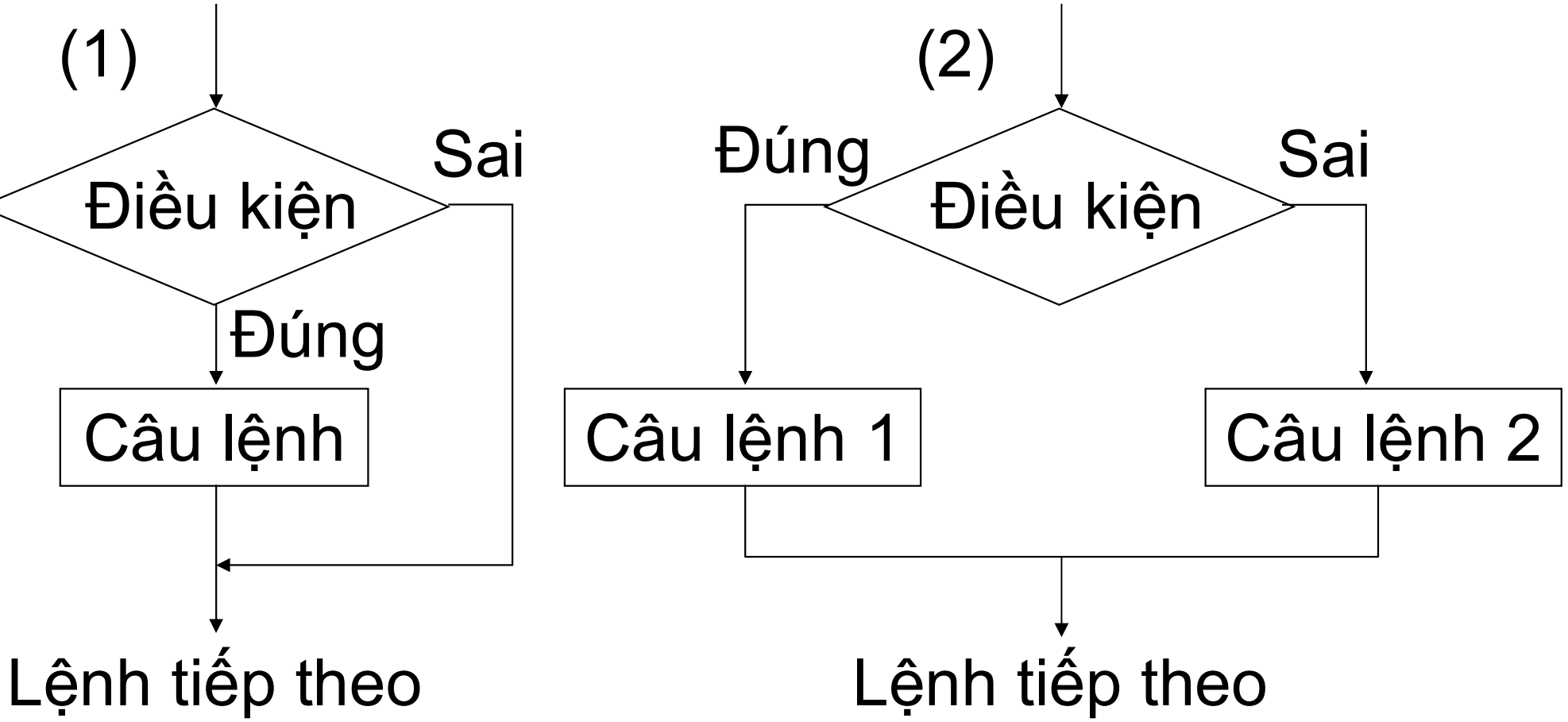
  (1) if (điều kiện) Câu lệnh;

  (2) if (điều kiện) Câu_lệnh_1; else Câu_lệnh_2;

  trong đó Câu_lệnh có thể là một câu lệnh đơn lẻ hoặc một khối lệnh. Lưu ý là Điều kiện phải đặt trong ngoặc và sau Câu_lệnh_1 vẫn phải có dấu chấm phẩy.
- Lệnh kiểm tra điều kiện là để bảo máy kiểm tra một điều kiện, nếu đúng thì làm công việc này, nếu sai thì làm công việc khác. Biểu thức điều kiện là một biểu thức logic có giá trị đúng (khác 0) hoặc sai (bằng 0).

1. Lệnh kiểm tra điều kiện if (tiếp)

✧ Lưu đồ thực hiện lệnh dạng (1) và (2) như sau:

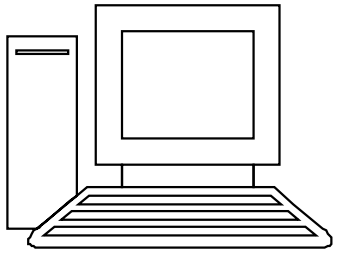


1. Lệnh kiểm tra điều kiện if (tiếp)

✧ Ví dụ 5.1: vdp1c51.cpp

Viết chương trình nhập vào một số thực, kiểm tra nếu số đó lớn hơn hoặc bằng 0 thì đưa ra màn hình căn bậc 2 của số đó, nếu âm thì đưa ra thông báo "Số âm không có căn bậc 2".

```
//Khai bao su dung thu vien chuong trinh
#include<iostream.h>
#include<math.h>
int main()
{
  float a;

  cout<<"Nhap vao mot so: ";
  cin>>a;
  if (a>=0) cout<<"Can bac 2 bang: "<<sqrt(a);
  else cout<<"So am khong tinh duoc can bac 2";
  return 0;
}
```



2. Lệnh thử và rẽ nhánh switch

- Khi cần kiểm tra giá trị của một biểu thức xem có bằng một giá trị nào trong nhiều giá trị không ta dùng lệnh switch.
- Cú pháp: có 2 dạng

(1)

```
switch (Biểu thức)
{
        case hằng1:
                Các câu lệnh;
                break;
        case hằng2:
                Các câu lệnh;
                break;
        ……
        case hằngN:
                Các câu lệnh;
                break;
}
```

Không có chấm phẩy

Các lệnh ứng với hằng 1

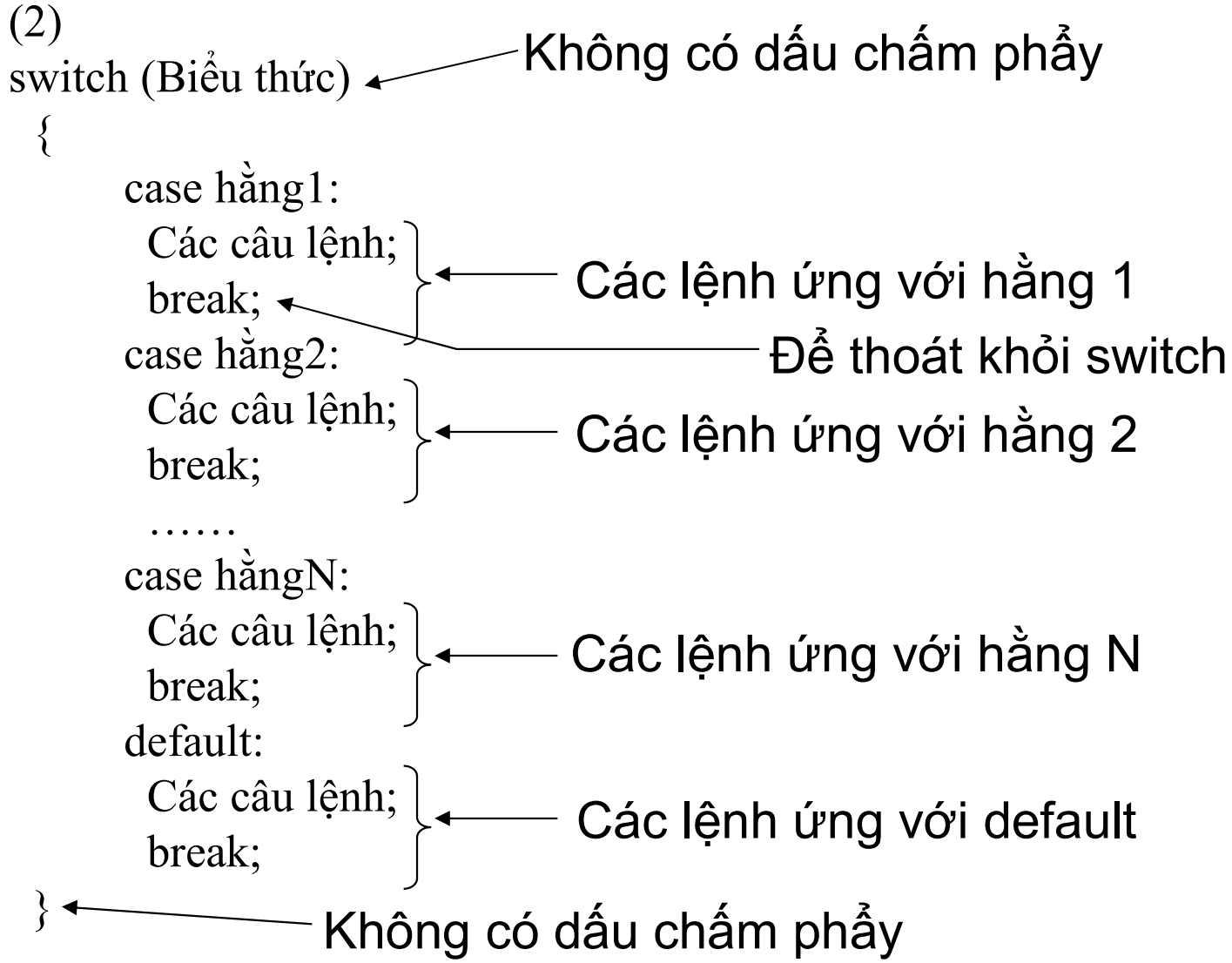
Để thoát khỏi switch

Các lệnh ứng với hằng 2

Các lệnh ứng với hằng N

Không có chấm phẩy

2. Lệnh thử và rẽ nhánh switch (tiếp)

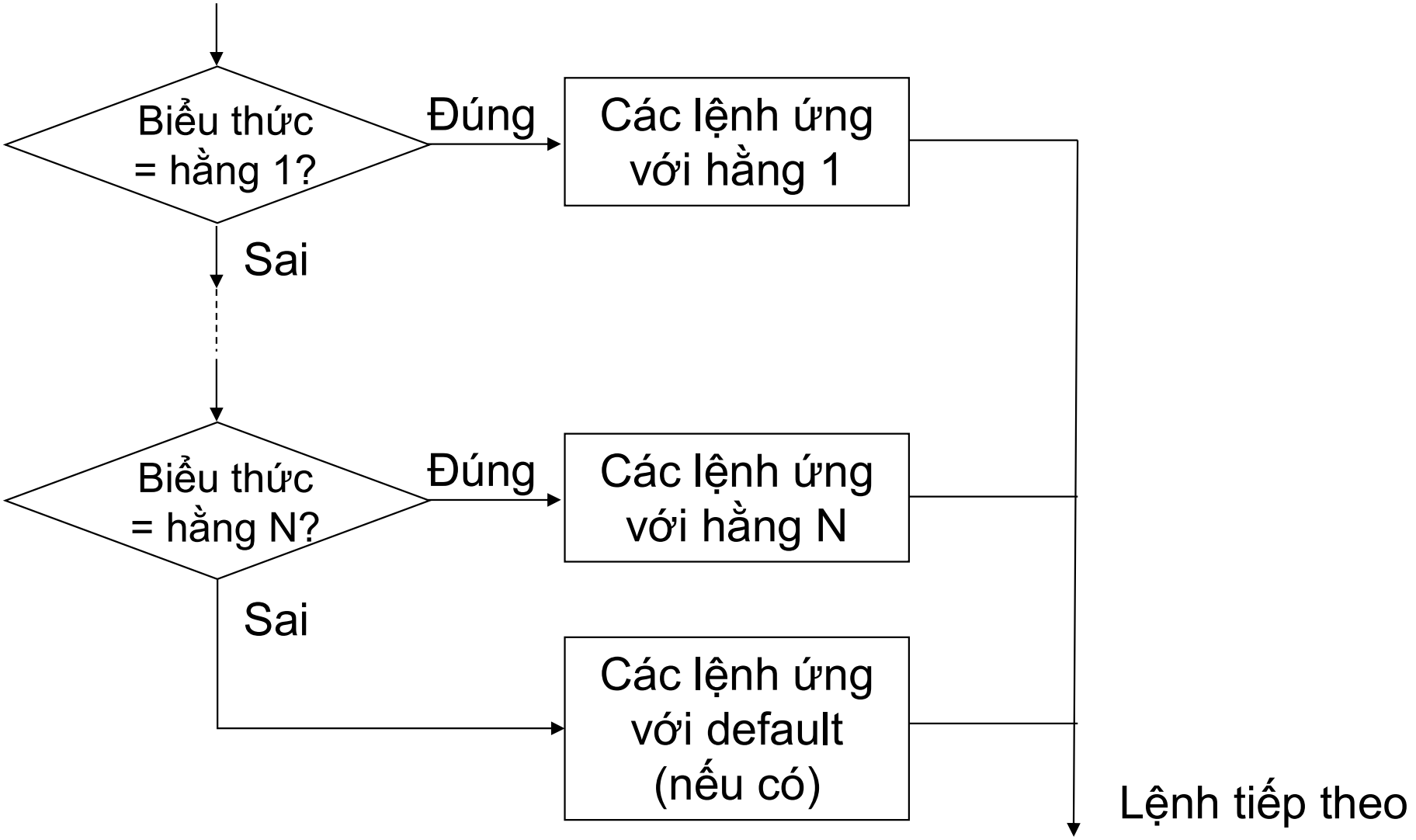


2. Lệnh thử và rẽ nhánh switch (tiếp)

- Biểu thức sau từ khoá switch phải đặt trong ngoặc đơn.
- **Biểu thức và các hằng phải cùng kiểu và phải là kiểu số nguyên hoặc ký tự.**
- Các hằng có thể là một giá trị hằng hoặc biểu thức hằng (các hằng kết hợp với nhau). Sau các hằng phải có dấu hai chấm.
- Trước mỗi hằng phải có từ khoá case, tức là không thể có nhiều hằng chung một từ khoá case.
- *Nếu muốn nhiều hằng cùng chung một câu lệnh thì các hằng này để gần nhau và chỉ viết các lệnh cùng câu lệnh break ở hằng dưới cùng.*

2. Lệnh thử và rẽ nhánh switch (tiếp)

Lưu đồ thực hiện lệnh switch như sau:



2. Lệnh thử và rẽ nhánh switch (tiếp)

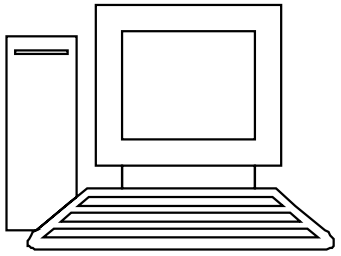
Ví dụ 5.2: vdp1c52.cpp

Viết chương trình nhập vào tháng và năm dương lịch, cho biết tháng trong năm đó có bao nhiêu ngày?

(*Chương trình trang sau)*

2. Lệnh thử và rẽ nhánh switch (tiếp)

```
//Chuong trinh vdp1c52.cpp
//Khai bao su dung thu vien chuong trinh
#include<iostream.h>
int main()
{
  int t,n;
  cout<<"Nhap vao thang: ";cin>>t;
  cout<<"Nhap vao nam: ";cin>>n;
  switch(t)
  {
      case 1:
      case 3:
      case 5:
      case 7:
      case 8:
      case 10:
      case 12:
        cout<<"Thang nay co 31 ngay";
        break;
      case 4:
      case 6:
      case 9:
      case 11:
        cout<<"Thang nay co 30 ngay";
        break;
      case 2:
        if(n%4==0 && n%100 != 0) cout<<"Thang nay co 29 ngay";
        else cout<<"Thang nay co 28 ngay";
        break;
  }

  return 0;
}
```



II. Lệnh lặp

1. Lệnh lặp với số lần lặp xác định for

2. Lệnh lặp với lần lặp không xác định

1. Lệnh lặp với số lần xác định for

- ✧ Để bảo máy thực hiện nhiều lần một số lệnh nào đó với số lần thực hiện xác định ta dùng lệnh lặp for.
- ✧ Cú pháp:

for (Biểu thức khởi tạo;Biểu thức kiểm tra; Biểu thức tăng/giảm)

Câu lệnh hoặc Khối lệnh

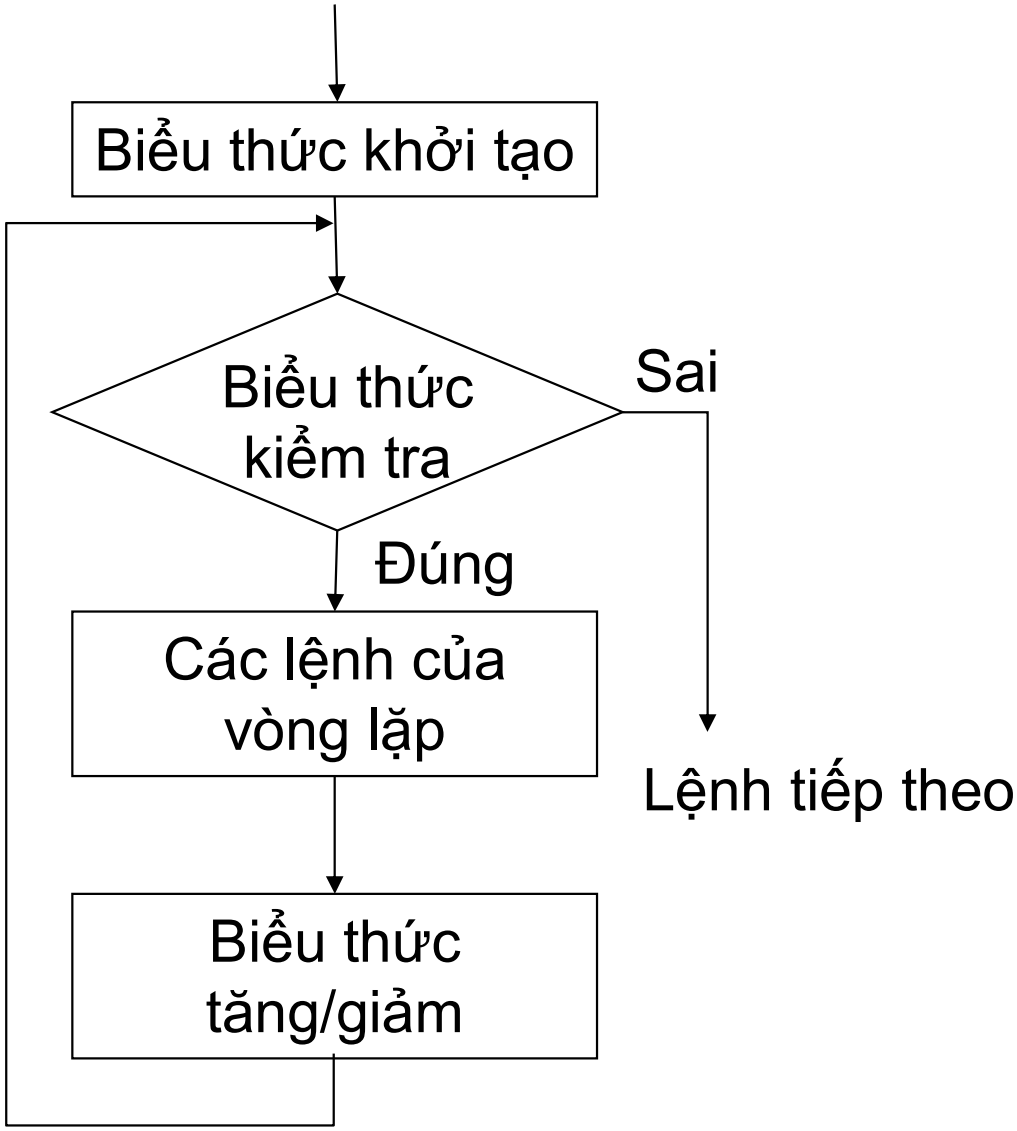
- Biểu thức khởi tạo dùng để khởi tạo giá trị ban đầu cho biến điều khiển vòng lặp và chỉ được thực hiện duy nhất một lần khi bắt đầu vào vòng lặp for. Trong biểu thức khởi tạo có thể khai báo và khởi tạo biến điều khiển, tuy nhiên biến điều khiển khai báo ở đây sẽ mất khi vòng lặp for kết lúc.

1. Lệnh lặp với số lần xác định for (tiếp)

- Biểu thức kiểm tra dùng để kiểm tra giá trị của biến điều khiển xem còn tiếp tục lặp hay kết thúc. Biểu thức kiểm tra thường là biểu thức logic có giá trị đúng hoặc sai, khi có giá trị đúng thì vẫn lặp, khi có giá trị sai thì kết thúc.
- Biểu thức tăng/giảm dùng để thay đổi biến điều khiển theo chiều tăng hoặc giảm.

1. Lệnh lặp với số lần xác định for (tiếp)

- Lưu đồ thực hiện lệnh for như bên:
- Ba biểu thức trong lệnh for có thể không có nhưng hai dấu chấm phẩy không thể thiếu. Khi không viết biểu thức kiểm tra thì mặc định biểu thức kiểm tra có giá trị true, điều này làm cho vòng lặp lặp mãi.



1. Lệnh lặp với số lần xác định for (tiếp)

✧ *Ví dụ:*

```
for (i=1;i<=10;i++)
    cout<<i<<'\n';
for (int j=10;j<=20;j+=2)
{
    cout<<j;
    cout<<'\n';
}
```

1. Lệnh lặp với số lần xác định for (tiếp)

Ví dụ: Tính tổng S = 1 + 2 + 3 + … + N

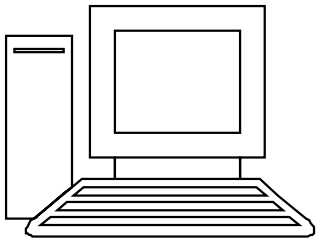
BTVN: 1) Viết chương trình tính gần đúng số π theo công thức sau (với n số hạng đầu tiên):

$$\frac{\pi}{4} = 1 - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + \ldots + \frac{(-1)^n}{2n+1}$$

2) Tính n!

1. Lệnh lặp với số lần xác định for (tiếp)

```
//Khai bao su dung thu vien chuong trinh
#include<iostream.h>
void main()
{
    int n,i;
    float s;

    cout<<"Nhap vao gia tri cua n: "; cin>>n;
    s=1;
    for(i=1;i<=n;i++)
        if(i%2 != 0) s-=1/(2*i+1);
        else s+=1/(2*i+1);

    cout<<"PI= "<<s*4;
}
```



2. Lệnh lặp với số lần lặp không xác định

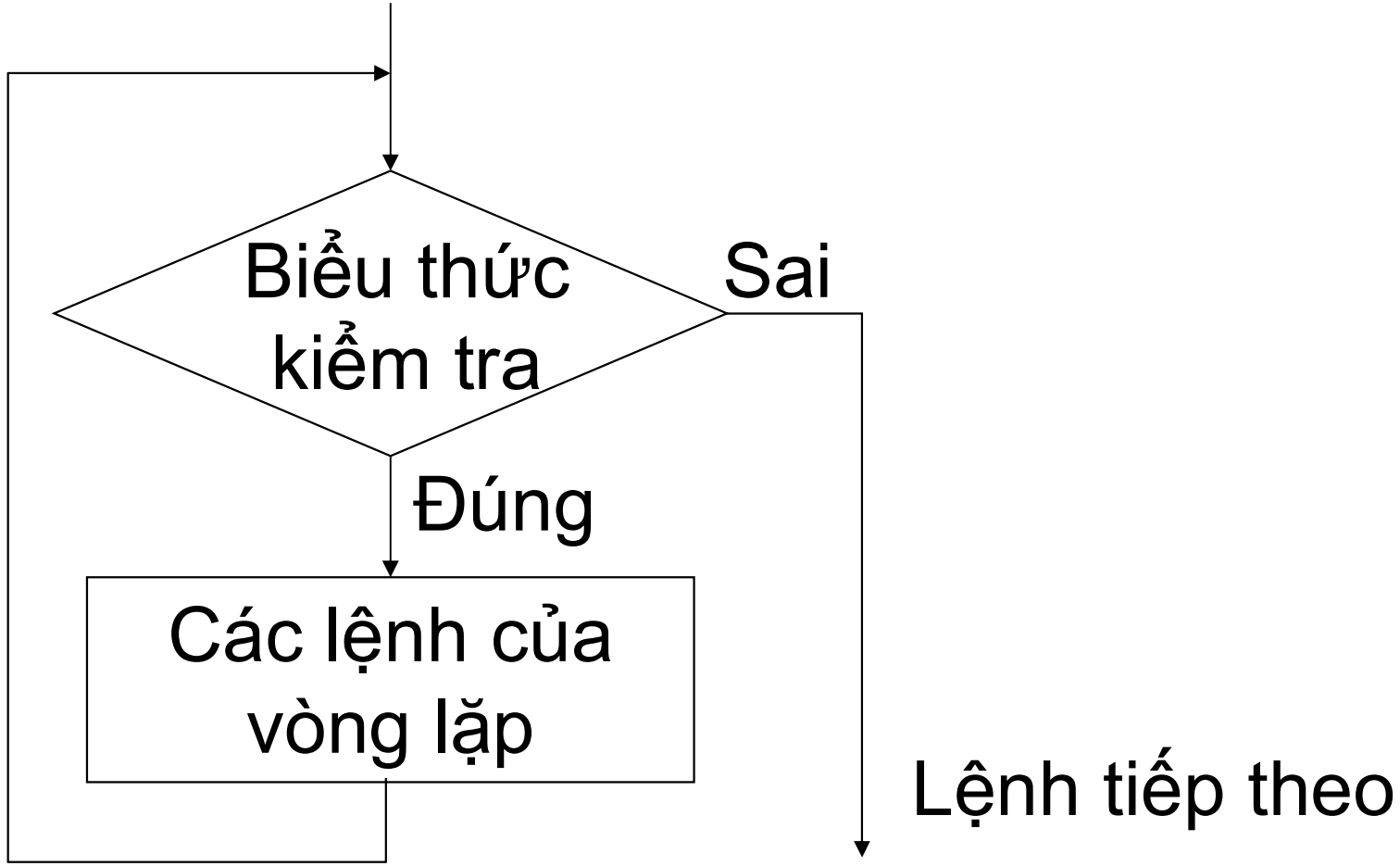
✧Lệnh lặp kiểm tra điều kiện trước while

```
while (Biểu thức kiểm tra)
    Câu lệnh;
```

← Không có dấu chấm phẩy

2. Lệnh lặp với số lần lặp không xác định (tiếp)

✧ Lưu đồ thực hiện lệnh while



2. Lệnh lặp với số lần lặp không xác định (tiếp)

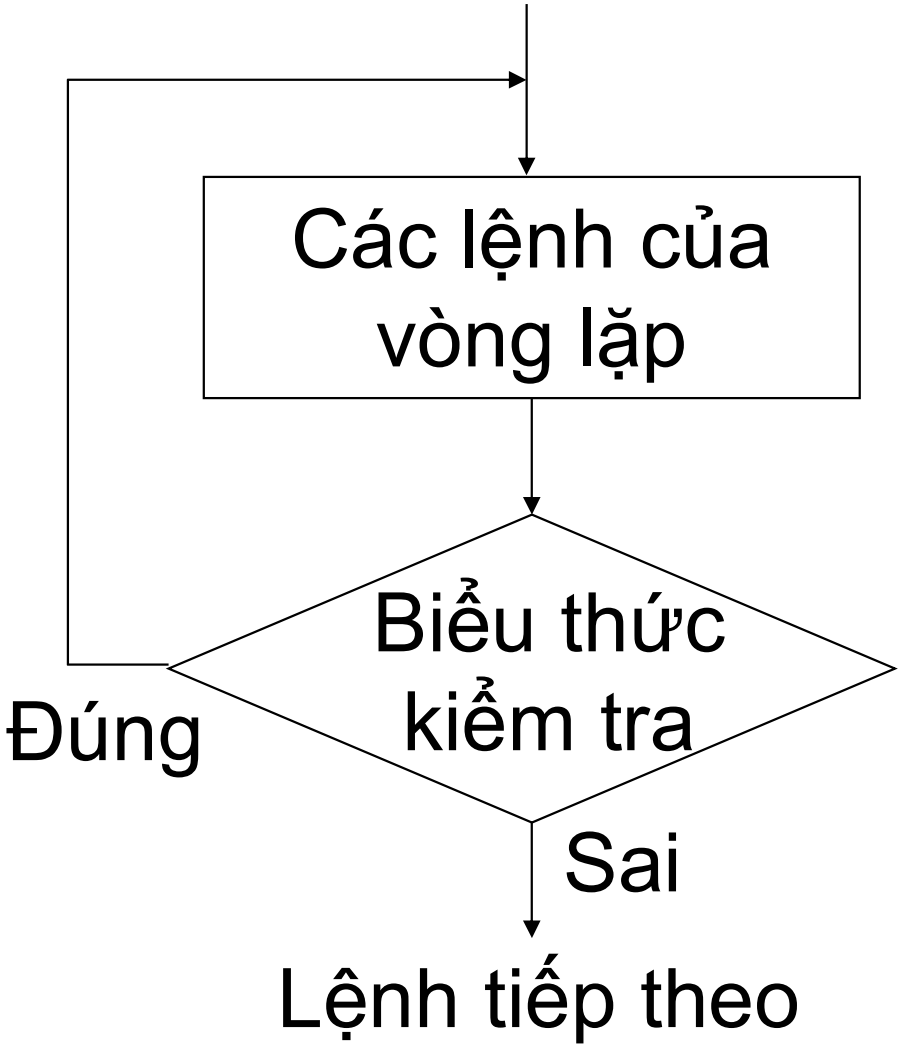
- Lệnh lặp kiểm tra điều kiện sau do-while

```
do
{
   Các câu lệnh;
}
while (Biểu thức kiểm tra);
```

Không có dấu chấm phẩy

2. Lệnh lặp với số lần lặp không xác định (tiếp)

✧ Lưu đồ thực hiện lệnh do … while



2. Lệnh lặp với số lần lặp không xác định (tiếp)

Ví dụ: Tìm USCLN(a,b)

BTVN: 1) Viết chương trình tính e^x theo công thức:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \ldots + \frac{x^n}{n!} + \ldots$$

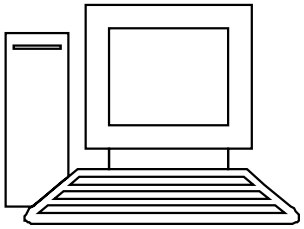
Với độ chính xác 0.0001, tức là ta cần chọn n sao cho

$$\left|\frac{x^n}{n!}\right| < 0.0001$$

2) Làm lại bài tính gần đúng số PI với độ chính xác là 10^{-4}.

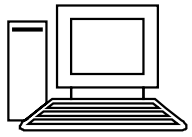
2. Lệnh lặp với số lần lặp không xác định (tiếp)

```
//Khai bao su dung thu vien chuong trinh
#include<iostream.h>
#include<math.h>

void main()
{
  int i;
  float x,s,tg;

  cout<<"Nhap vao gia tri cua x: "; cin>>x;
  s=1;tg=1;i=1;
  do
  {
    tg*=x/i++;
    s+=tg;
  }
  while(fabs(tg)>=0.0001);

  cout<<"e mu "<<x<<"= "<<s;
}
```



III. Lệnh break

- Lệnh break được dùng để thoát khỏi lệnh for, while, do-while và switch. Nếu các lệnh này lồng nhau thì lệnh break thoát khỏi lệnh bên trong nhất chứa nó.
- Với lệnh break ta có thể thoát khỏi vòng lặp từ một điểm bất kỳ bên trong vòng lặp mà không dùng đến điều kiện kết thúc vòng lặp.
- *Ví dụ:* Viết chương trình nhập vào một số nguyên dương, cho biết số này có phải là số nguyên tố không?

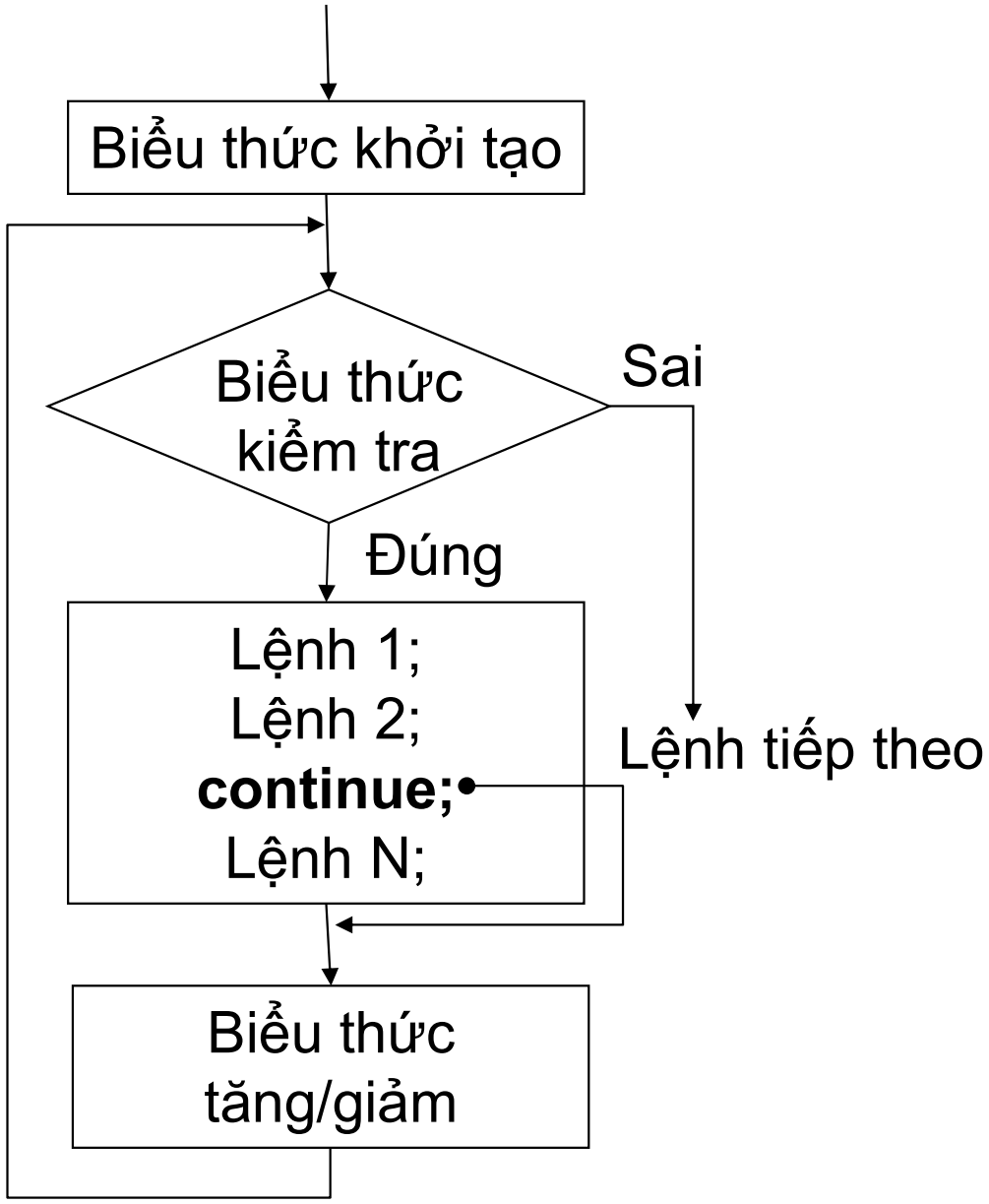


IV. Lệnh continue

- Lệnh continue chỉ dùng với các lệnh lặp for, while và do-while.
- Lệnh continue không làm thoát khỏi lệnh lặp mà làm cho lệnh lặp bỏ qua các lệnh sau lệnh continue để thực hiện vòng lặp tiếp theo.
- Tác động của lệnh continue với các lệnh lặp được làm rõ qua các lưu đồ thực hiện lệnh dưới đây.

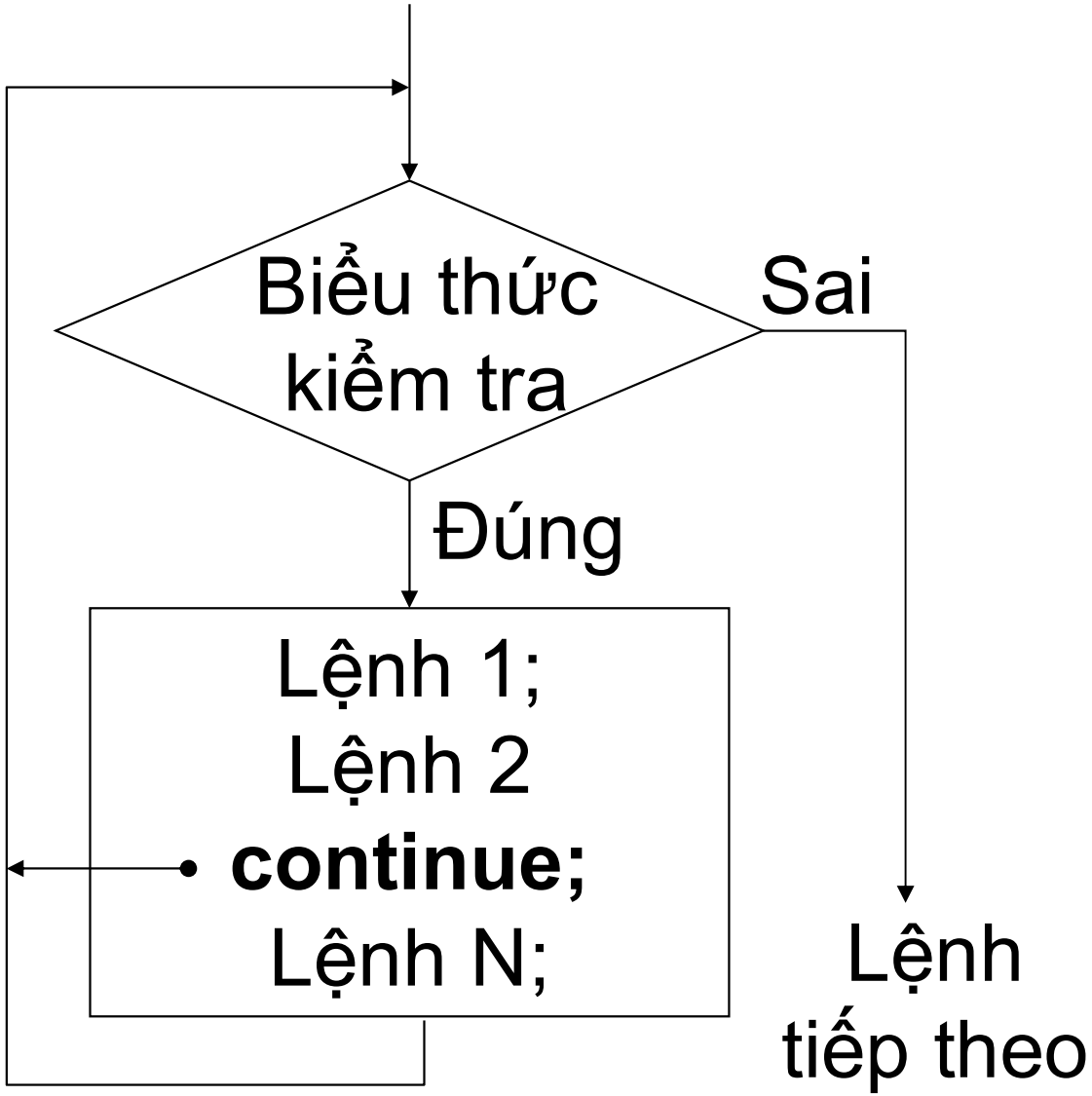
IV. Lệnh continue (tiếp)

- Tác động của lệnh continue đối với lệnh for.



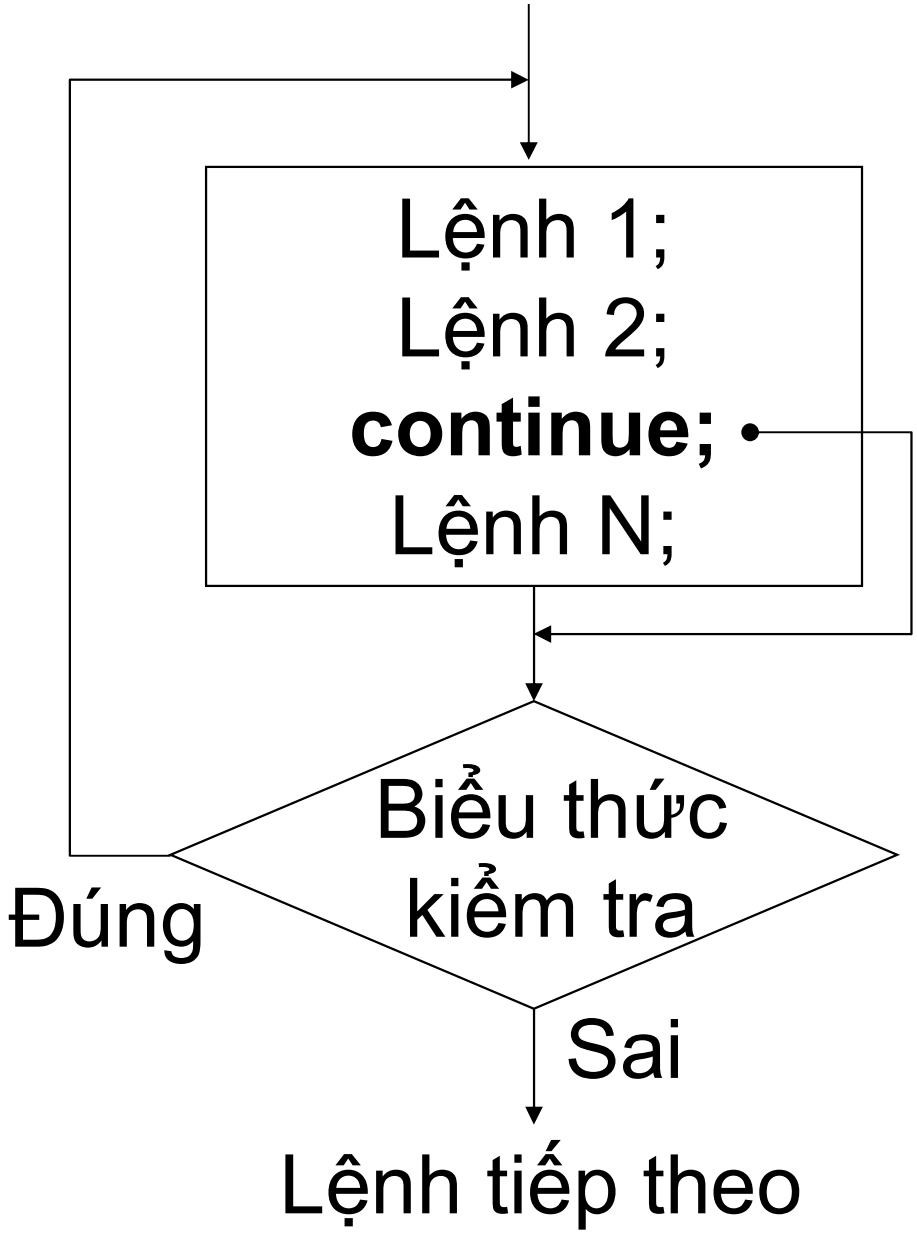
IV. Lệnh continue (tiếp)

- Tác động của lệnh continue đối với lệnh while.



IV. Lệnh continue (tiếp)

- Tác động của lệnh continue đối với lệnh do-while.



Bài tập

1) Viết chương trình tính sin$x$ với độ chính xác 0.0001 theo công thức:

$$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} + \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!}$$