Câu lệnh điều khiển

KHOA CÔNG NGHỆ THÔNG TIN
ĐẠI HỌC MỞ TPHCM

Mục tiêu

- Biết cách sử dụng kết quả của biểu thức luận lý lựa chọn chuỗi câu lệnh được thực hiện.
- Biết cách sử dụng lệnh **`if-else`** hoặc **`switch`** để giải quyết trường hợp có nhiều lựa chọn.
- Biết cách sử dụng câu lệnh lặp **`while`**, **`do-while`** hoặc **`for`** để thực hiện lặp lại các câu lệnh trong chương trình.
- Biết cách sử dụng các câu lệnh lặp lồng vào nhau.
- Biết cách sử dụng **`break`** và **`continue`** trong các câu lệnh lặp
- Có khả năng phân tích vấn đề, sử dụng các câu lệnh thích hợp để viết một chương trình hoàn chỉnh.

Các bước giải quyết vấn đề lập trình

1. Phân tích (analysis):

   Xác định vấn đề. Mô tả các yêu cầu.

2. Thiết kế (design):

   Mô tả thuật giải giải quyết vấn đề.

3. ***Lập trình (programming):***

   Mô tả giải pháp bằng mã lệnh (chương trình).

4. Bộ thử nghiệm (testing).

Câu lệnh điều khiển

- Khái niệm câu lệnh
- Câu lệnh điều khiển rẽ nhánh
- Câu lệnh điều khiển lựa chọn
- Câu lệnh lặp
- Lệnh break, contiue

Khái niệm câu lệnh

- Khái niệm
- Khối lệnh
- Phân loại

Khái niệm câu lệnh

- Câu lệnh/Phát biểu (statement) là một:
  - khai báo hoặc
  - biểu thức kết thúc bằng dấu chấm phẩy **;** hoặc
  - khối lệnh đặt giữa hai dấu **{** và **}** hoặc
  - "câu lệnh điều khiển".
- Ví dụ:

```
int a;
a = 5;
if (a == b)
  cout << "Hai so bang nhau";
else
  cout << "Hai so khong bang nhau";
…
```

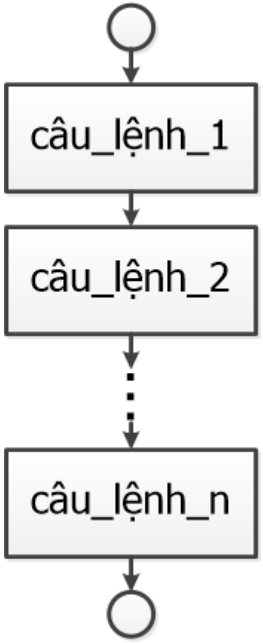
- Câu lệnh rỗng (empty statement): chỉ có dấu chấm phẩy **;**

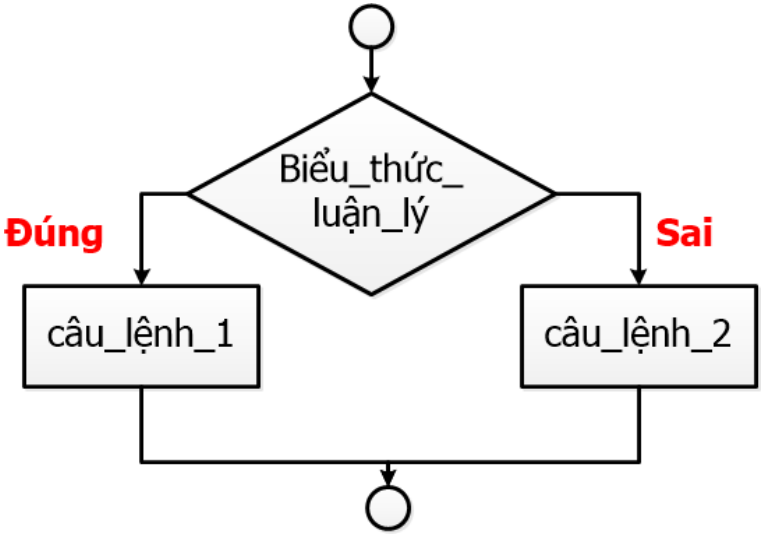
Khối lệnh

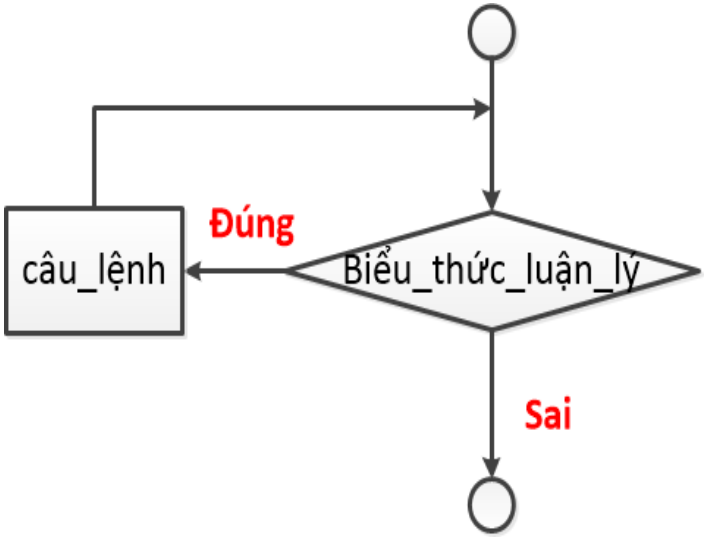
- Khối lệnh (block): gồm các câu lệnh được đặt giữa hai dấu ngoặc **{** và **}**

```
{
        Câu_lệnh_1;
        Câu_lệnh_2;
        …
        Câu_lệnh_n;
}
```

Phân loại







1. Lệnh tuần tự

2. Lệnh rẽ nhánh

3. Lệnh lặp

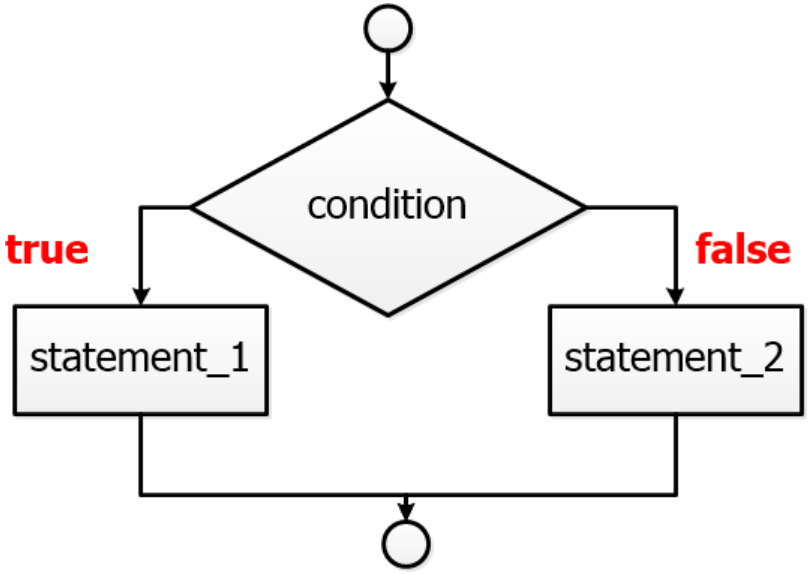
Câu lệnh điều khiển rẽ nhánh

- Cú pháp
- Ví dụ
- Một số lỗi thường gặp
- Cú pháp đa chiều
- Lưu ý
- Toán tử điều kiện

if-else

- **Cú pháp**

```
if (condition)
      statement_1;
[else
      statement_2;]
```



- Trong đó:
  - **condition** (điều kiện): là một biểu thức luận lý, trả về giá trị đúng (true) hoặc sai (false).
  - **statement_1** và **statement_2** (câu lệnh 1 và câu lệnh 2)**:** có thể là lệnh rỗng, lệnh đơn hoặc khối lệnh.
  - Dấu **[** và **]** cho biết nội dung bên trong có thể khuyết.

if-else

- **Ví dụ 1:**

Viết chương trình nhận vào điểm trung bình 1 môn học của một sinh viên. Xuất ra màn hình thông báo cho biết sinh viên này đậu hay rớt? Biết rằng nếu điểm trung bình từ 5 trở lên là đậu, ngược lại là rớt.

Phân tích (analysis):

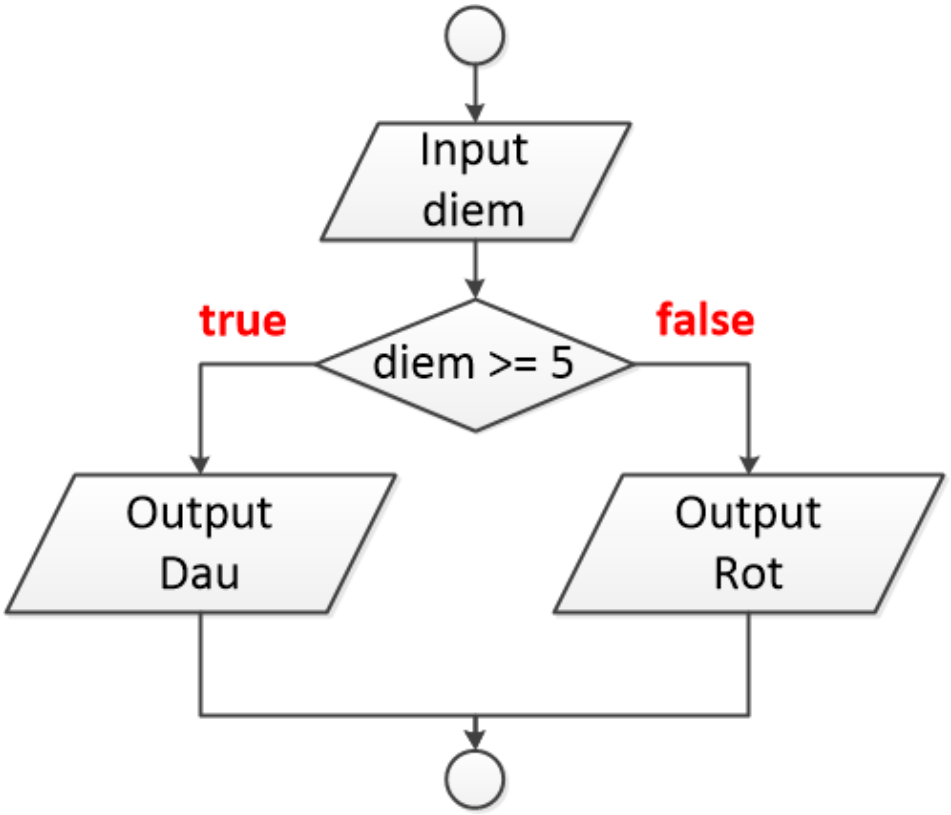
- Input: **diem** (điểm trung bình)
- Output: xuất thông báo **Dau / Rot**

if-else

- **Ví dụ 1:**

Thiết kế (design)

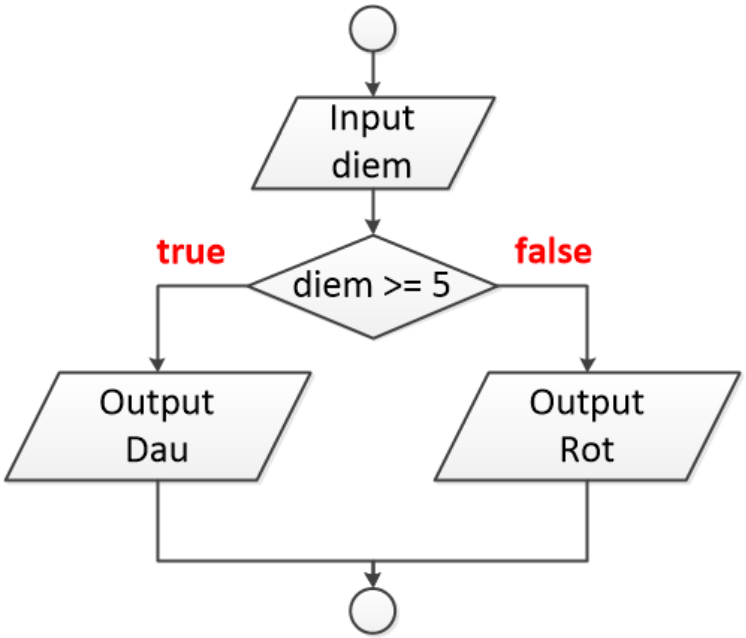
- Nhập **`diem`**
- Nếu **`diem >= 5`** đúng thì xuất **`Dau`**
- Ngược lại thì xuất **`Rot`**



if-else

- **Ví dụ 1:**

Thiết kế (design)



Lập trình (programming)

```
#include <iostream>
using namespace std;
int main()
{
        double diem;
        cout << "Nhap diem: ";
        cin >> diem;
        if (diem >= 5)
           cout << "Dau\n";
        else
           cout << "Rot\n";
        return 0;
}
```

if-else

- **Ví dụ 1:**

Kiểm thử (testing)

diem	Kết quả
7.2	Dau
5	Dau
4.9	Rot

Lập trình (programming)

```
#include <iostream>
using namespace std;
int main()
{
      double diem;
      cout << "Nhap diem: ";
      cin >> diem;
      if (diem >= 5)
         cout << "Dau\n";
      else
         cout << "Rot\n";
      return 0;
}
```

if-else

- **Ví dụ 2:**

Viết chương trình tìm số lớn nhất của 2 số nguyên a và b.

Phân tích (analysis):

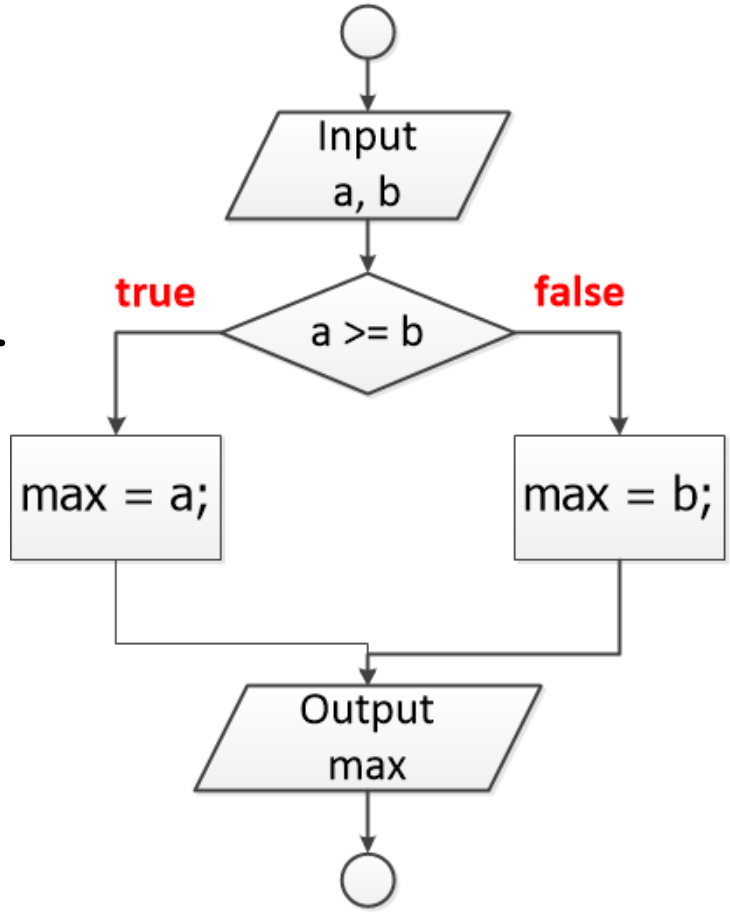
- Input: 2 số nguyên a và b
- Output: xuất thông báo max (số lớn nhất của a và b)

if-else

- **Ví dụ 2:**

Thiết kế (design)

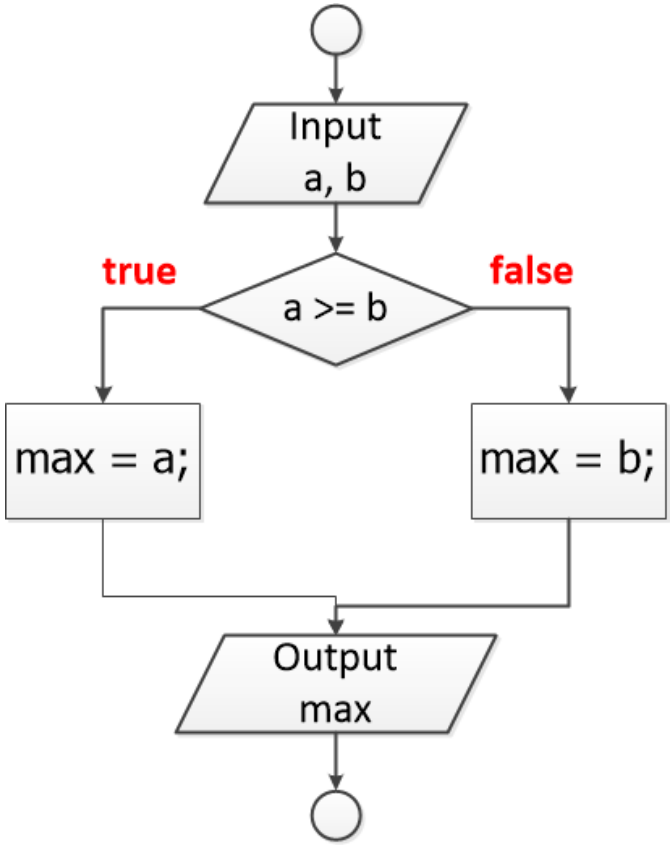
- Nhập 2 số nguyên a, b
- Nếu **`a >= b`** đúng thì **`max = a`**
- Ngược lại thì **`max = b`**
- Xuất max



if-else

- **Ví dụ 2:**

Thiết kế (design)



Lập trình (programming)

```
#include <iostream>
using namespace std;
int main()
{
    int  a, b, max;
    cout << "Nhap so thu nhat: ";
    cin >> a;
    cout << "Nhap so thu hai: ";
    cin >> b;
    if (a >= b)
       max = a;
    else
       max = b;
    cout << "So lon nhat la: " <<
    max << endl;
    return 0;
}
```

if-else

- **Ví dụ 2:**

Kiểm thử (testing)

a	b	Kết quả
5	2	So lon nhat la 5
5	5	So lon nhat la 5
4	9	So lon nhat la 9

Lập trình (programming)

```
#include <iostream>
using namespace std;
int main()
{
    int  a, b, max;
    cout << "Nhap so thu nhat: ";
    cin >> a;
    cout << "Nhap so thu hai: ";
    cin >> b;
    if (a >= b)
       max = a;
    else
       max = b;
    cout << "So lon nhat la: " <<
    max << endl;
    return 0;
}
```

if-else

- **Ví dụ 3:**

Viết chương trình tìm giá trị tuyệt đối của một số nguyên n.

Phân tích (analysis):

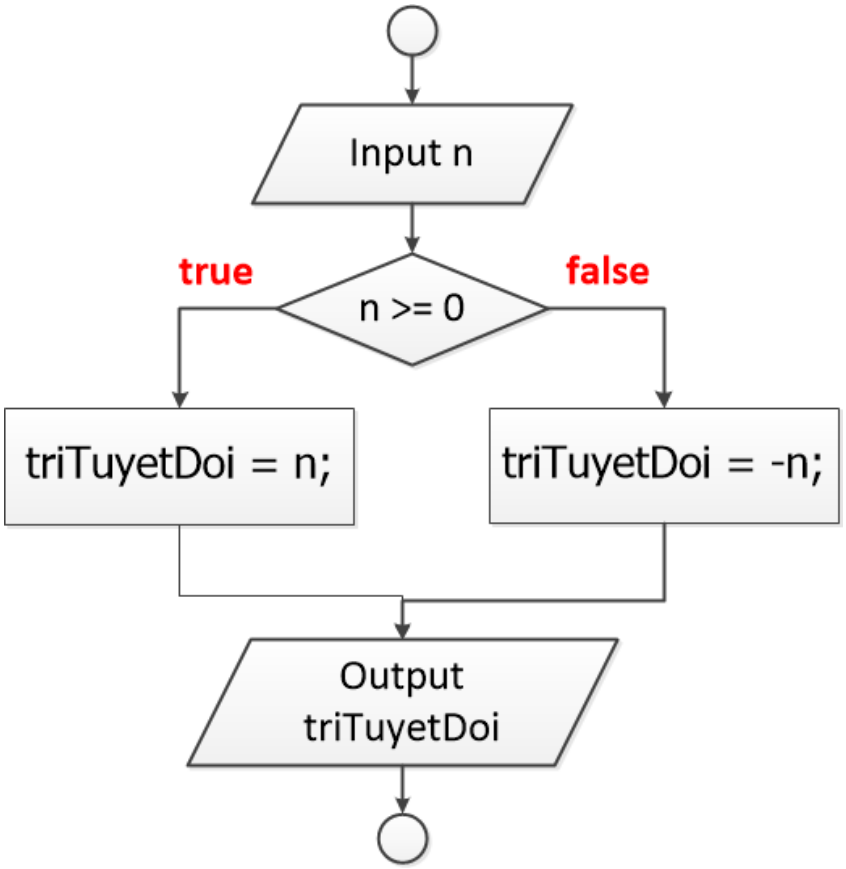
- Input: số nguyên n
- Output: kết quả trị tuyết đối của n

if-else

- **Ví dụ 3 – cách 1**

Thiết kế (design)

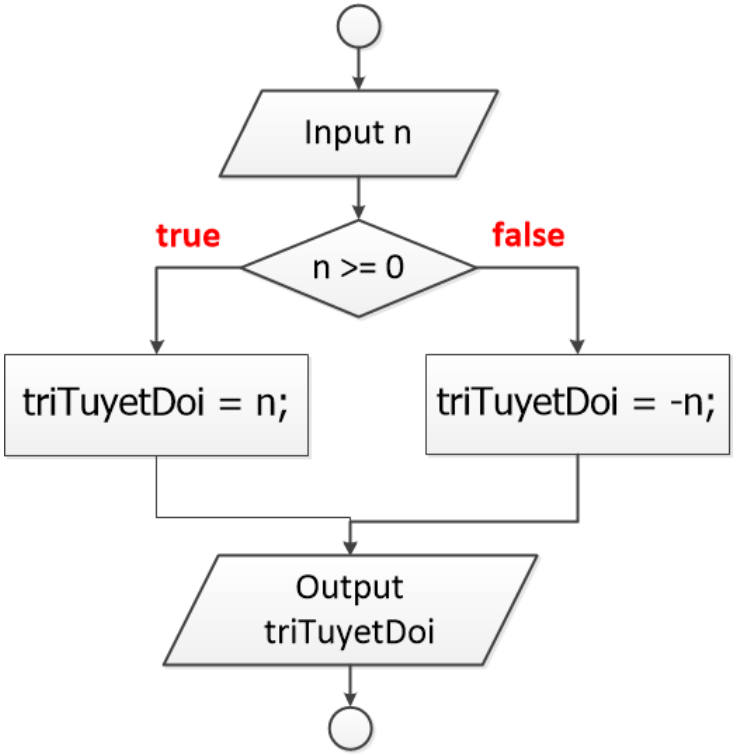
- Nhập số nguyên n
- Nếu n >= 0 thì triTuyetDoi = n;
- Ngược lại thì triTuyetDoi = -n;
- Xuất triTuyetDoi



if-else

- **Ví dụ 3 – cách 1**

Thiết kế (design)



Lập trình (programming)

```
#include <iostream>
using namespace std;
int main()
{
    int  n, triTuyetDoi;
    cout << "Nhap so nguyen: ";
    cin >> n;
    if (n >= 0)
       triTuyetDoi = n;
    else
       triTuyetDoi = -n;
    cout << "Ket qua tri tuyet
    doi cua " << n << "la: " <<
    triTuyetDoi << endl;
    return 0;
}
```

if-else

- **Ví dụ 3 – cách 1**

Kiểm thử (testing)

n	|n|
5	5
0	0
-4	4

Lập trình (programming)

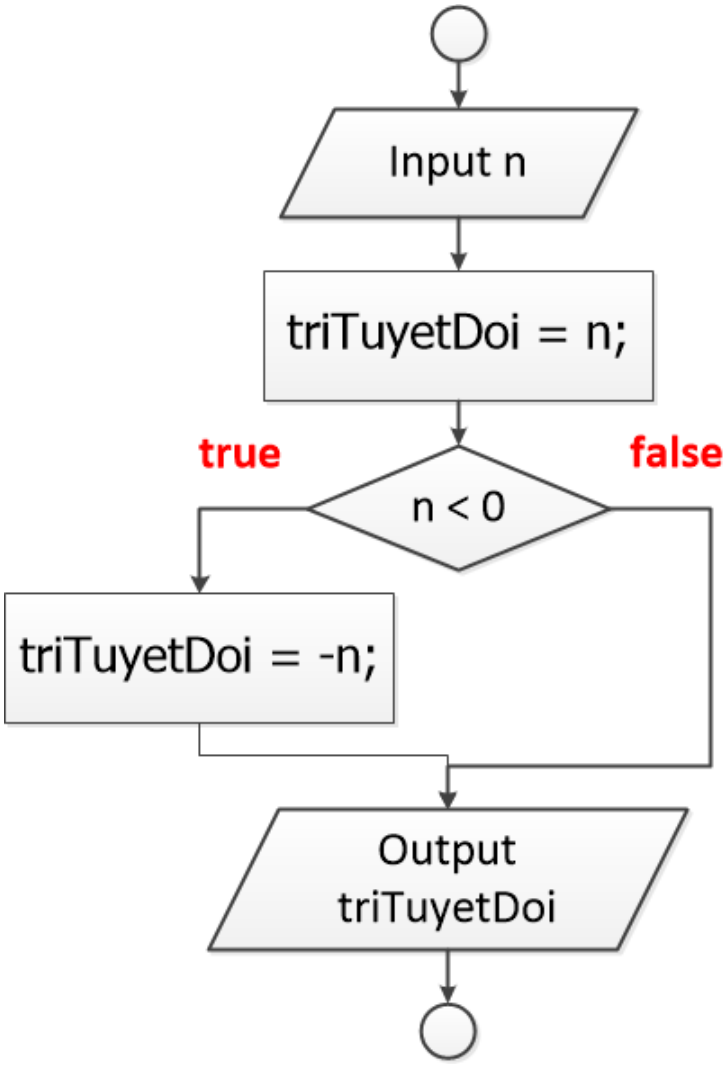
```
#include <iostream>
using namespace std;
int main()
{
    int  n, triTuyetDoi;
    cout << "Nhap so nguyen: ";
    cin >> n;
    if (n >= 0)
       triTuyetDoi = n;
    else
       triTuyetDoi = -n;
    cout << "Ket qua tri tuyet
    doi cua " << n << "la: " <<
    triTuyetDoi << endl;
    return 0;
}
```

if-else

- **Ví dụ 3 – cách 2**

Thiết kế (design)

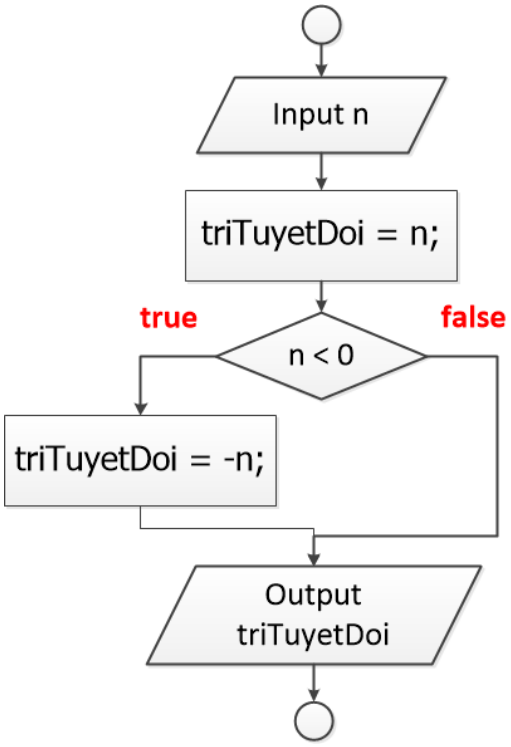
- Nhập số nguyên n
- Gán triTuyetDoi = n;
- Nếu n < 0 thì triTuyetDoi = -n;
- Xuất triTuyetDoi



if-else

- **Ví dụ 3 – cách 2**

Thiết kế (design)



Lập trình (programming)

```
#include <iostream>
using namespace std;
int main()
{
    int  n, triTuyetDoi;
    cout << "Nhap so nguyen: ";
    cin >> n;
    triTuyetDoi = n;
    if (n < 0)
       triTuyetDoi = -n;
    cout << "Ket qua tri tuyet
    doi cua " << n << "la: " <<
    triTuyetDoi << endl;
    return 0;
}
```

if-else

- **Một số lỗi thường gặp:**
  - Trong biểu thức luận lý thường nhầm lẫn phép so sánh (dấu ==) với phép gán (dấu =)

Ví dụ:

```
if (x = y)
    cout << "x va y bang nhau" << endl;
```

//Khi biên dịch không báo lỗi (error) hay cảnh báo (warning), luôn luôn mặc định giá trị của biểu thức luận lý là đúng (true)

if-else

- **Một số lỗi thường gặp:**
  - Gõ dấu **;** ngay sau biểu thức luận lý vì như vậy lệnh ngay sau đó sẽ luôn thực hiện dù biểu thức luận lý có giá trị là sai (false).

Ví dụ:

```
if (x == y) ;
    cout << "x va y bang nhau" << endl;
```

//Màn hình khi biên dịch hiển thị lỗi cảnh báo:

```
warning C4390: ';' : empty controlled
statement found; is this the intent?
```

//Giả sử x = 3 và y = 4 thì câu x và y bằng nhau vẫn được xuất ra màn hình

if-else

- **Một số lỗi thường gặp:**
  - Gõ biểu thức luận lý sau mệnh đề else.

Ví dụ:

```
if (x == y)
   cout << "x va y bang nhau" << endl;
else (x != y)
   cout << "x va y khong bang nhau" << endl;
```

//Khi biên dịch màn hình thông báo lỗi cú pháp

Cú pháp if-else lồng nhau

- **Cú pháp 1:**

```
if (condition)
    statement;
else
   if (condition)
     statement;
   else
     if (condition)
        statement;
        …
        else
           statement;
```

Cú pháp if-else đa chiều

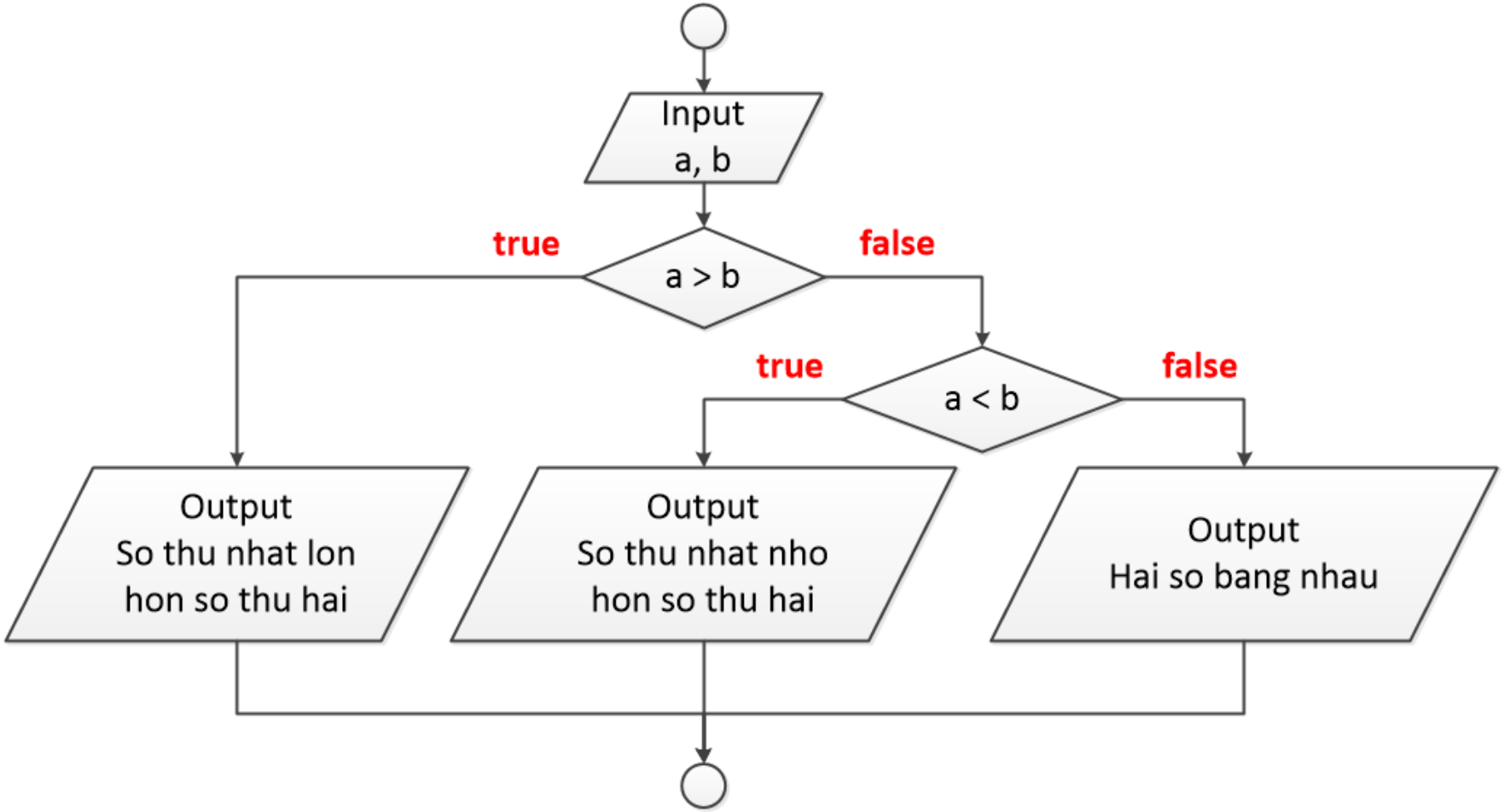
- **Ví dụ 4:**

Viết chương trình nhận vào 2 số nguyên. Xuất ra màn hình kết quả so sánh giữa hai số (số thứ nhất lớn hơn, nhỏ hơn hay hai số bằng nhau)

Input	Processing	Output
2 số nguyên a, b	- Nhập a, b - Nếu a > b đúng thì xuất "**so thu nhat lon hon so thu hai**" - Ngược lại *(ngầm hiểu a > b là sai)* thì so sánh nếu a < b đúng thì xuất "**so thu nhat nho hon so thu hai**" - Ngược lại *(ngầm hiểu a > b sai và a < b cũng sai)* thì xuất "**hai so bang nhau**"	Xuất kết quả so sánh

Cú pháp if-else lồng nhau

- **Ví dụ 4:**



Cú pháp if-else lồng nhau

- **Ví dụ 4:**

```
#include <iostream>
using namespace std;
int main()
{
    int a, b;
    cout << "Nhap so nguyen thu nhat: ";
    cin >> a;
    cout << "Nhap so nguyen thu hai: ";
    cin >> b;
    if (a > b)
       cout << "So thu nhat lon hon so thu hai\n";
    else
       if (a < b)
          cout << "So thu nhat nho hon so thu hai\n";
       else
          cout << "Hai so bang nhau\n";
    return 0;
}
```

Cú pháp if-else lồng nhau

- **Ví dụ 4:**

a	b	Kết quả xuất ra màn hình
5	2	So thu nhat lon hon so thu hai
3	4	So thu nhat nho hon so thu hai
6	6	Hai so bang nhau

Cú pháp if-else lồng nhau

- **Cú pháp 2:**

```
if (condition)
   statement;
else if (condition)
   statement;
else if (condition)
   statement;
   …
else
   statement;
```

Câu lệnh rẽ nhánh

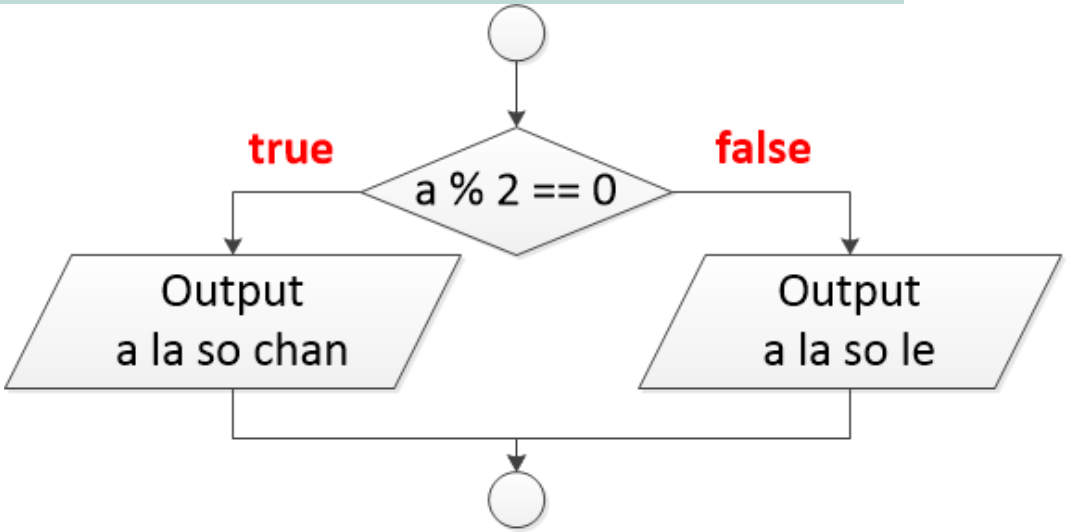
- **Lưu ý:**
  - Lệnh điều khiển rẽ nhánh có thể dùng để kiểm tra dữ liệu nhập vào có hợp lệ hay không.

```
double diem;
cout << "Nhap diem: ";
cin >> diem;
if (diem >= 0 && diem <= 10)
{
    //diem hop le
    //thuc hien cac viec tinh toan khi diem hop le va xuat ket qua
}
else
    cout << "Nhap diem khong hop le!";
```

Câu lệnh rẽ nhánh

- **Lưu ý:**
  - Nếu cùng 1 yêu cầu tính toán, lệnh if-else sẽ giải quyết nhanh hơn liên tục nhiều lệnh if khuyết else.

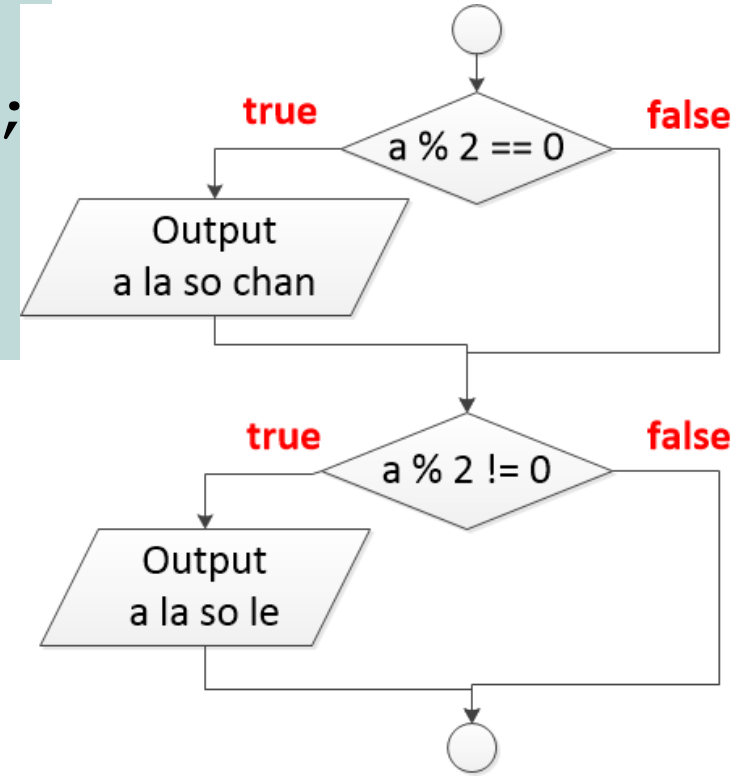
```
if (a % 2 == 0)
  cout << "a la so chan\n";
else
  cout << "a la so le\n";
```



Câu lệnh rẽ nhánh

- **Lưu ý:**
  - Nếu cùng 1 yêu cầu tính toán, lệnh if-else sẽ giải quyết nhanh hơn liên tục nhiều lệnh if khuyết else.

```
if (a % 2 == 0)
   cout << "a la so chan\n";
if (a % 2 != 0)
   cout << "a la so le\n";;
```



Câu lệnh rẽ nhánh

- **Lưu ý:**
  - Khi sử dụng if..else lồng nhau, nếu không có khối lệnh ngăn cách bởi cặp dấu **{** và **}** thì trình biên dịch luôn hiểu **else** sẽ là trường hợp ngược lại của **if gần nhất**.

```
if (x == 2)
   if (y == 4)
      cout << "x bang 2 va y bang 4" << endl;
   else
      cout << "x khac 2" << endl;
```

//Nếu nhập x = 2, y = 5 thì sẽ xuất câu x khac 2
//Nếu nhập x = 3, y = 4 thì không xuất gì cả

Câu lệnh rẽ nhánh

- **Lưu ý:**
  - Khi sử dụng if..else lồng nhau, nếu không có khối lệnh ngăn cách bởi cặp dấu **{** và **}** thì trình biên dịch luôn hiểu **else** sẽ là trường hợp ngược lại của **if gần nhất**.

```
if (x == 2)
{
   if (y == 4)
      cout << "x bang 2 va y bang 4" << endl;
}
else
   cout << "x khac 2" << endl;
```

//Nếu nhập x = 2, y = 5 thì sẽ không xuất gì cả
//Nếu nhập x = 3, y = 4 thì xuất x khac 2

Toán tử điều kiện

- Toán tử điều kiện: mang ý nghĩa tương đương với if-else.
  - **Cú pháp:**

```
condition ? expression_1 :  expression_2;
```

- Trong đó:
  - **condition**: điều kiện, là một biểu thức luận lý (true/false)
  - Nếu **condition** có giá trị **true** thì sẽ thực hiện tính toán và trả về kết quả **expression_1** (biểu thức 1).
  - Nếu **condition** có giá trị **false** thì sẽ thực hiện tính toán và trả về kết quả **expression_2** (biểu thức 2).

Toán tử điều kiện

- **Ví dụ:** chương trình tìm số lớn nhất của 2 số nguyên a và b.

```
#include <iostream>
using namespace std;
int main()
{
   int a,  b;
   int max;
   cout << "Nhap so thu nhat: "; cin >> a;
   cout << "Nhap so thu hai: "; cin >> b;
   max = a >= b ? a : b;
   cout << "So lon nhat la: " << max <<
   endl;
}
//Nếu nhập a = 3,  b = 5 thì kết quả max là 5
//Nếu nhập a = 7, b = 2 thì kết quả max là 7
```

Câu lệnh lựa chọn switch

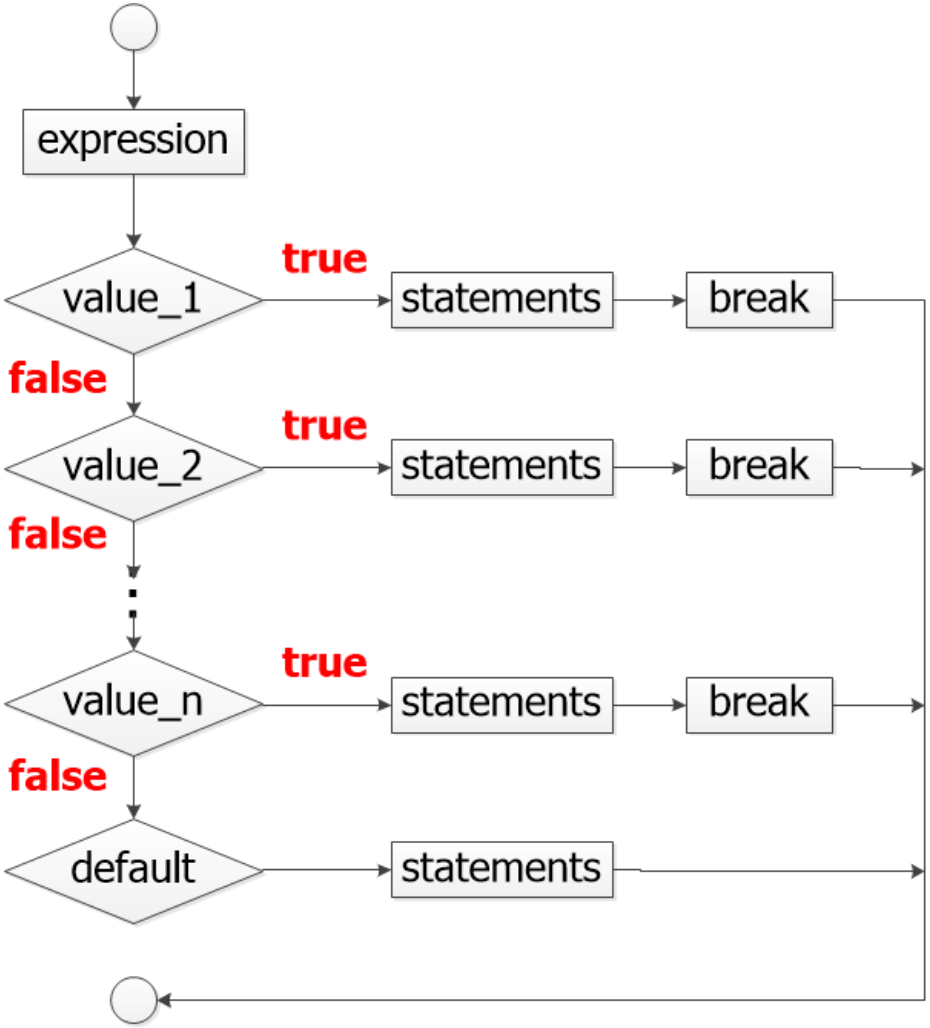
- Cú pháp
- Ví dụ
- Một số lưu ý

switch

- **Cú pháp**
  - Giá trị của **expression** có kiểu: bool, char hoặc int.
  - Mỗi nhãn **case** có một giá trị (**value**).
  - **value** là hằng và có cùng kiểu dữ liệu với **expression**.
  - Câu lệnh **break** dùng thoát cấu trúc switch.
  - Nhãn **default** có thể khuyết.

```
switch(expression)
{
   case value_1:
      statements;
      break;
   case value_2:
     statements;
      break;
      ...
   case value_n:
      statements;
      break;
  [default:
      statements;]
}
```

switch




```
switch(expression)
{
   case value_1:
      statements;
      break;
   case value_2:
     statements;
      break;
      ...
   case value_n:
      statements;
      break;
  [default:
      statements;]
}
```

switch

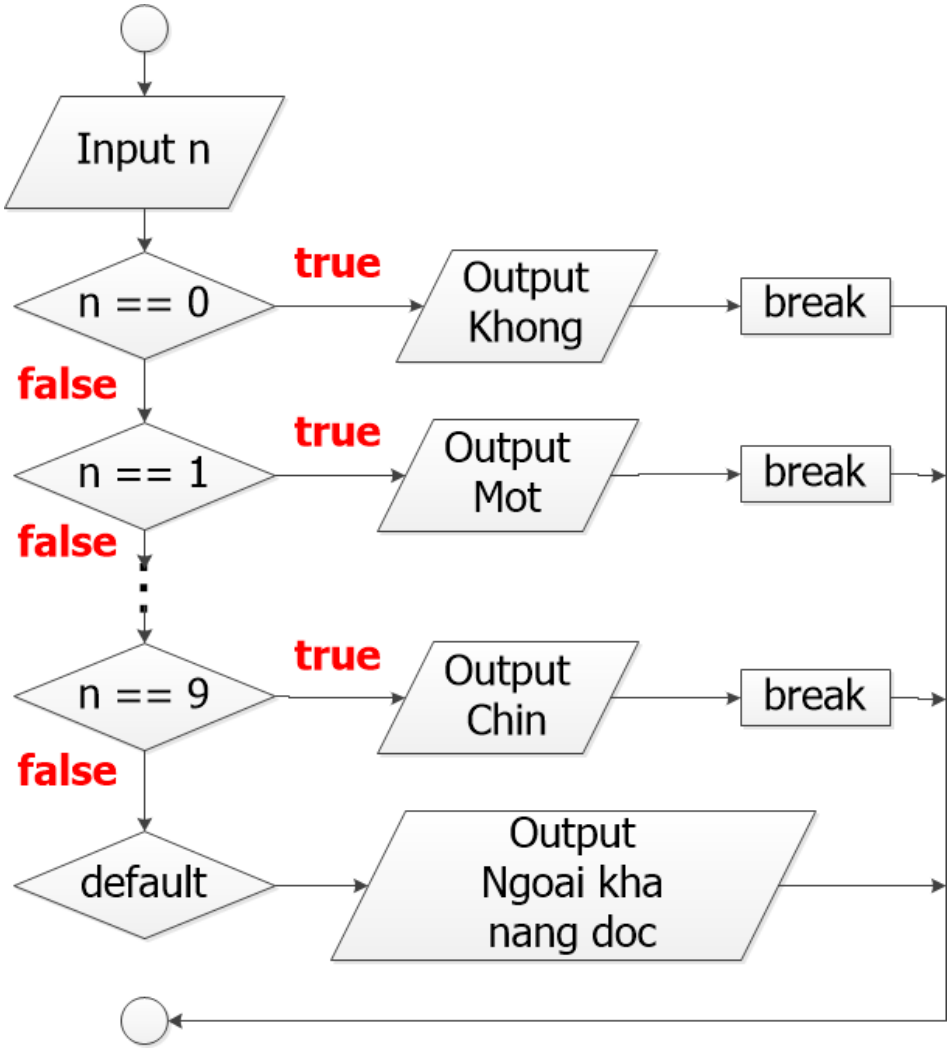
- **Ví dụ:**

Viết chương trình nhận vào một số từ 0 đến 9. Xuất ra màn hình đọc số đó dưới dạng chữ, nếu ngoài phạm vi từ 0 đến 9 thì xuất thông báo ngoài khả năng đọc.

Input	Processing	Output
Số nguyên từ 0 đến 9	- Nhập số nguyên n - Nếu n == 0 đúng thì xuất "Khong" và kết thúc. Nếu sai thì đi tiếp. - Nếu n == 1 đúng thì xuất "Mot" và kết thúc. Nếu sai đi tiếp. - … - Nếu n == 9 đúng thì xuất "Chin" và kết thúc. - Nếu n == 9 sai thì xuất "Ngoai kha nang doc" và kết thúc.	Xuất kết quả đọc chữ

switch

- **Ví dụ:**


Input n
n == 0
true
Output Khong
break
false
n == 1
true
Output Mot
break
false
n == 9
true
Output Chin
break
false
default
Output Ngoai kha nang doc

switch

- **Ví dụ:**

```
#include <iostream>
using namespace std;
int main()
{
    int so;
    cout << "Nhap 1 so
    nguyen tu 0 den 9: ";
    cin >> so;
    switch(so)
    {
    case 0:
        cout << "Khong\n";
        break;
    case 1:
        cout << "Mot\n";
        break;
    case 2:
        cout << "Hai\n";
        break;
    …
    case 9:
        cout << "Chin\n";
        break;
    default:
        cout << "Ngoai kha nang
        doc\n";
    }
    return 0;
}
```

switch

- **Một số lưu ý**
  - **`switch`** không làm việc với kiểu **`string`**
  - Các **`case`** không được trùng giá trị (**`value`**).
  - Lệnh **`break`**; đặt sau mỗi trường hợp để kết thúc đoạn lệnh **`switch`** đó mà không thực hiện các trường hợp còn lại.

switch

- **Một số lưu ý**
  - **Ví dụ:**

```
switch(so)
{
case 0:
    cout << "Khong";
    break;
case 1:
    cout << "Mot";
    break;
case 2:
    cout << "Hai";
    break;
…
case 9:
    cout << "Chin";
    break;
default:
   cout << "Ngoai kha nang doc";
}
//nếu so là 1 thì xuất MotHai
```

switch

- **Một số lưu ý**
  - Có thể tận dụng việc bỏ qua lệnh **`break;`** ở cuối để tiếp tục thực hiện lệnh ở trường hợp kế tiếp.

```
char kt;
cout << "nhap ky tu: ";
cin >> kt;
switch (kt)
{
    case 'a': case 'o': case 'i': case 'u':
case 'e':
      cout << "la ky tu nguyen am\n";
      break;
    default:
      cout << "khong la ky tu nguyen am\n";
}
```

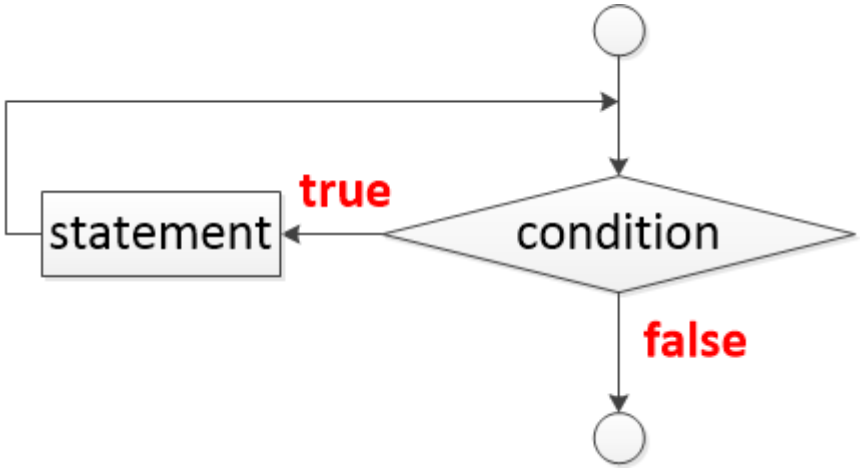
Cấu trúc lặp

- Lệnh while
- Lệnh do-while
- Lưu ý khối lệnh trong while và do-while
- Lệnh for
- Một số lưu ý

while

- **Cú pháp**

```
while (condition)
    statement;
```



- Trong đó:
  - **condition** (điều kiện): là biểu thức luận lý, có giá trị true/false.
  - **statement** (câu lệnh): có thể là lệnh rỗng, lệnh đơn hay khối lệnh.

while

- **`condition`** (điều kiện) với **`while`** gọi là điều kiện lặp (**`loop condition`**).
  - Nếu điều kiện lặp là **`true`** thì sẽ thực hiện hết các lệnh (**`statement`**).
  - Nếu điều kiện lặp là **`false`** thì sẽ ngưng và kết thúc lệnh lặp.
- Điều kiện lặp trong tất cả các câu lệnh lặp được tạo bằng cách:
  - Dùng biến đếm (**`counter`**)
  - Dùng giá trị cầm canh (**`sentinel value`**)
  - Dùng biến cờ (**`flag`**)
  - Kiểm tra trạng thái (**`state`**)

while

- **Ví dụ 1 dùng biến đếm (`counter`)**

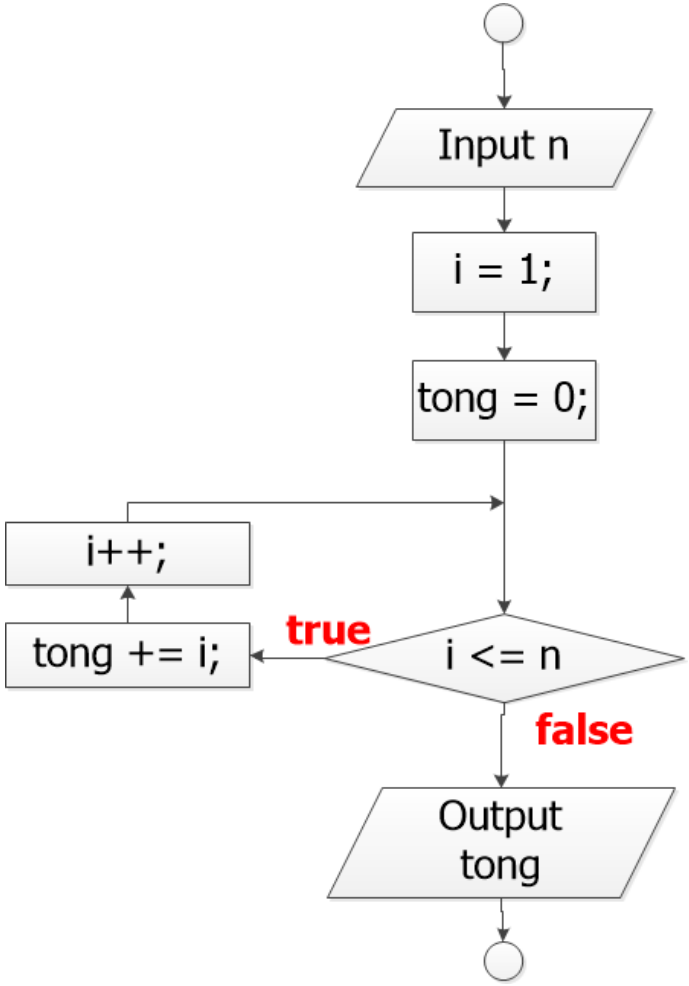
Viết chương trình nhập vào một số nguyên dương n.
Tính: S = 1 + 2 + 3 + … + n

Input	Processing	Output

Số nguyên dương n	• Nhập n
	• Khởi tạo "tổng tích lũy" tong = 0, "biến đếm" i = 1.
	• **Lặp lại** các lệnh sau đây **nếu i <= n**
	  • Cộng i vào tong: tong = tong + i;
	  • Tăng giá trị i lên 1 đơn vị: i++;
	• Xuất tong
		Xuất kết quả tổng từ 1 đến n

while

- **Ví dụ 1 dùng biến đếm (`counter`)**



while

- **Ví dụ 1 dùng biến đếm (counter)**

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    int i = 1;        //Khởi tạo biến đếm i = 1
    int tong = 0;     //Khởi tạo biến tong =0
    cout << "Nhap so nguyen duong n: ";
    cin >> n;
    while (i <= n)
    {
        tong += i;  //Cộng tích lũy kết quả vào biến tong
        i++;  //Tăng i lên 1 để lặp lại kiểm tra điều kiện lặp
    }
    cout << "Tong tu 1 den " << n << " la = " << tong
    << endl;
    return 0;
}
```

while

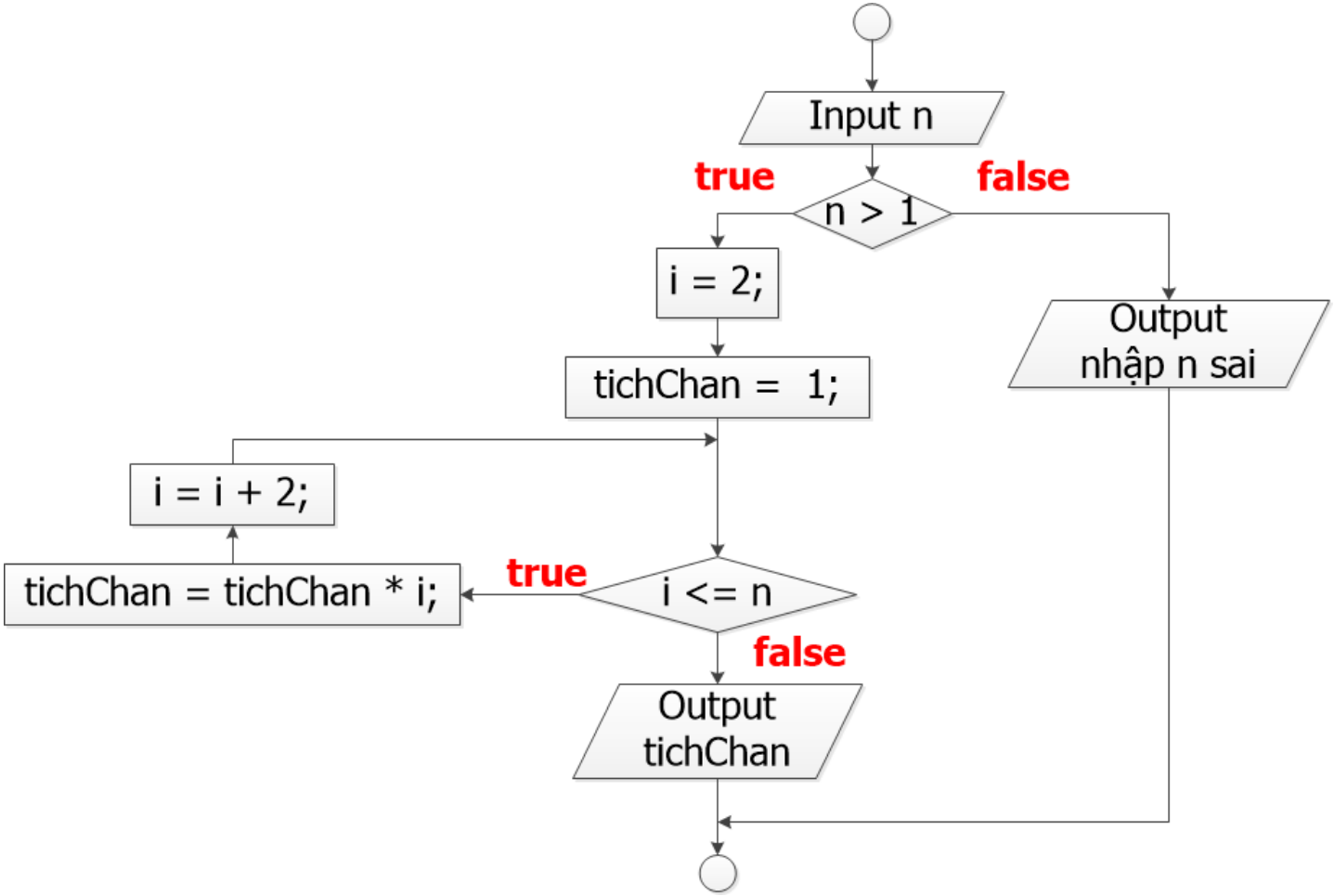
- **Ví dụ 2 dùng biến đếm (`counter`)**

Viết chương trình nhập vào một số nguyên dương n lớn hơn 1. Tính tích các số chẵn từ 1 đến n.

Input	Processing	Output
Số nguyên dương n	• Nhập n • Nếu n > 1 đúng thì tiếp tục thực hiện các bước còn lại, sai thì xuất thông báo nhập sai và kết thúc chương trình. • Khởi tạo i = 2; tichChan = 1; • **Lặp lại** các lệnh sau đây **nếu i <= n** - Tính tichChan = tichChan * i; - Tăng biến đếm i lên 2 đơn vị. • Xuất tichChan	Xuất kết quả tích các số chẵn từ 1 đến n

while

- **Ví dụ 2 dùng biến đếm (`counter`)**



while

- **Ví dụ 2 dùng biến đếm (counter)**

```
#include <iostream>
using namespace std;
int main()
{
    int n, i = 2, tichChan = 1;
    cout << "Nhap so nguyen duong n: ";
    cin >> n;
    if (n > 1)
    {
      while (i <= n)
      {
         tichChan *= i;
         i = i + 2;
      }
      cout << "Tich cac so chan tu 1 den " << n << " la = " <<
      tichChan << endl;
    }
    else
        cout << "Nhap n sai\n";
    return 0;
}
```

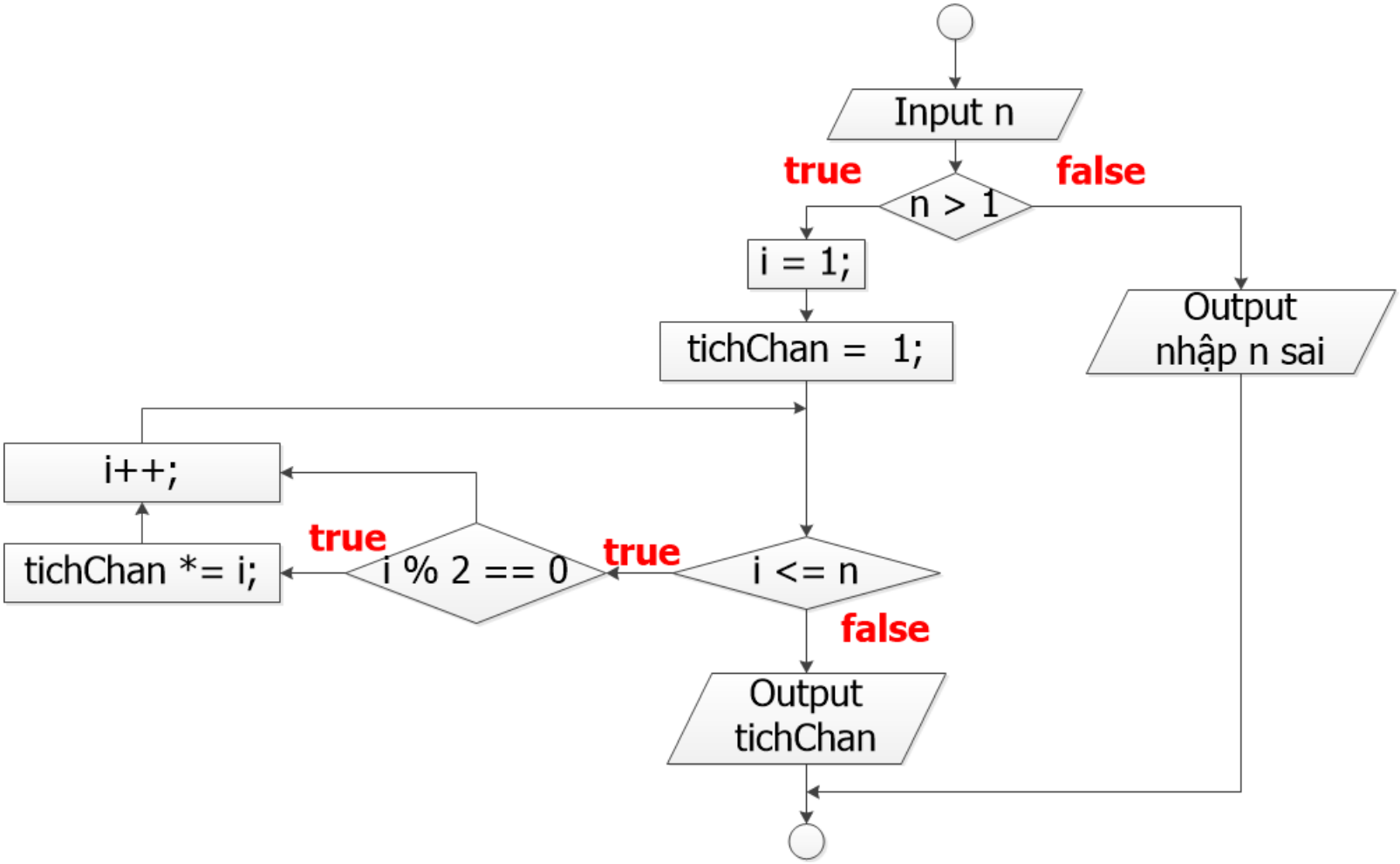
while

- **Ví dụ 2 dùng biến đếm (`counter`) – cách 2**

Input	Processing	Output
Số nguyên dương n	• Nhập n • Nếu n > 1 đúng thì tiếp tục thực hiện các bước còn lại, sai thì xuất thông báo nhập sai và kết thúc chương trình. • Khởi tạo i = 1; tichChan = 1; • **Lặp lại** các lệnh sau đây **nếu i <= n** - Nếu i % 2 dư 0 đúng thì Tính tichChan = tichChan * i; - Tăng biến đếm i lên 1 đơn vị. • Xuất tichChan	Xuất kết quả tích các số chẵn từ 1 đến n

while

- **Ví dụ 2 dùng biến đếm (`counter`) – cách 2**



while

- **Ví dụ 2 dùng biến đếm (`counter`) – cách 2**

```
#include <iostream>
using namespace std;
int main()
{
    int n, i = 1, tichChan = 1;
    cout << "Nhap so nguyen duong n: ";
    cin >> n;
    if (n > 1)
    {
      while (i <= n)
      {
         if (i%2==0)
            tichChan *= i;
         i++;
      }
      cout << "Tich cac so chan tu 1 den " << n << " la = " <<
      tichChan << endl;
    }
    else
        cout << "Nhap n sai\n";
    return 0;
}
```

while

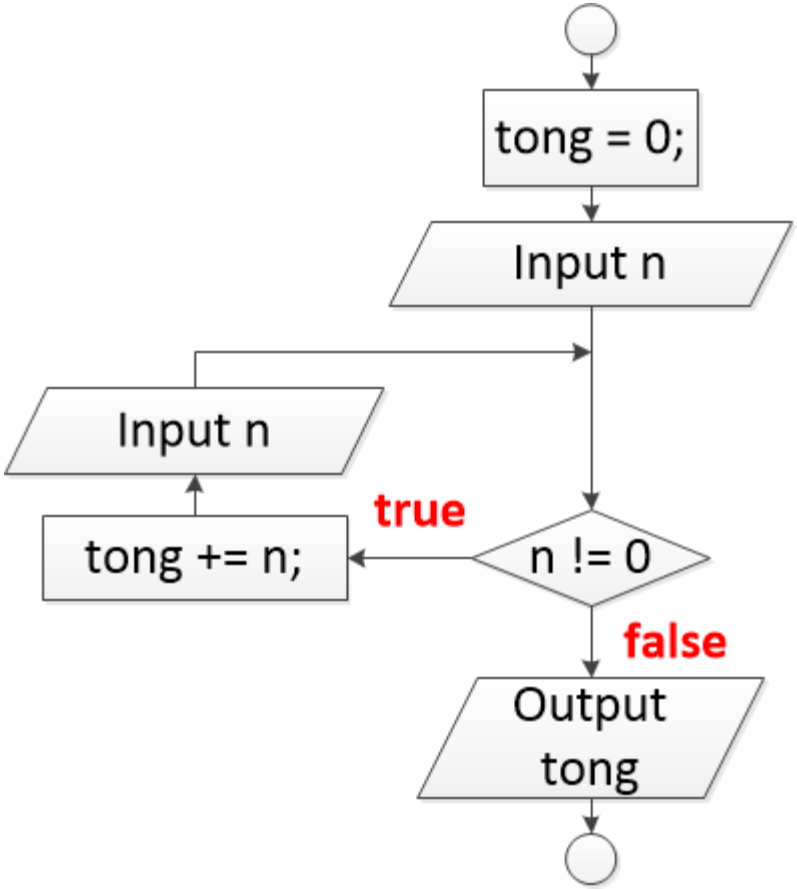
- **Ví dụ 3 dùng giá trị cầm canh (`sentinel value`)**

Viết chương trình tính tổng các số nguyên khác 0 được nhập vào, khi muốn kết thúc sẽ nhập 0.

Input	Processing	Output
Các số nguyên khác 0	• Khởi tạo biến tong = 0; • Nhập 1 số nguyên n • **Lặp lại** các bước sau nếu n != 0 ✓ Tính tong = tong + n; ✓ Nhập thêm giá trị khác cho n • Xuất tong	Xuất kết quả tổng các số nguyên khác 0 vừa nhập

while

- **Ví dụ 3 dùng giá trị cầm canh (`sentinel value`)**



while

- **Ví dụ 3 dùng giá trị cầm canh (`sentinel value`)**

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    int tong =  0;
    cout << "Nhap so nguyen (nhap 0 de ket thuc): ";
    cin >> n;
    while (n!=0)
    {
        tong += n;
        cout << "Nhap so nguyen (nhap 0 de ket thuc): ";
        cin >> n;
    }
    cout << "Tong cac so vua nhap la: " << tong <<
    endl;
    return 0;
}
```

while

- **Ví dụ 4 dùng giá trị cầm canh (`sentinel value`)**

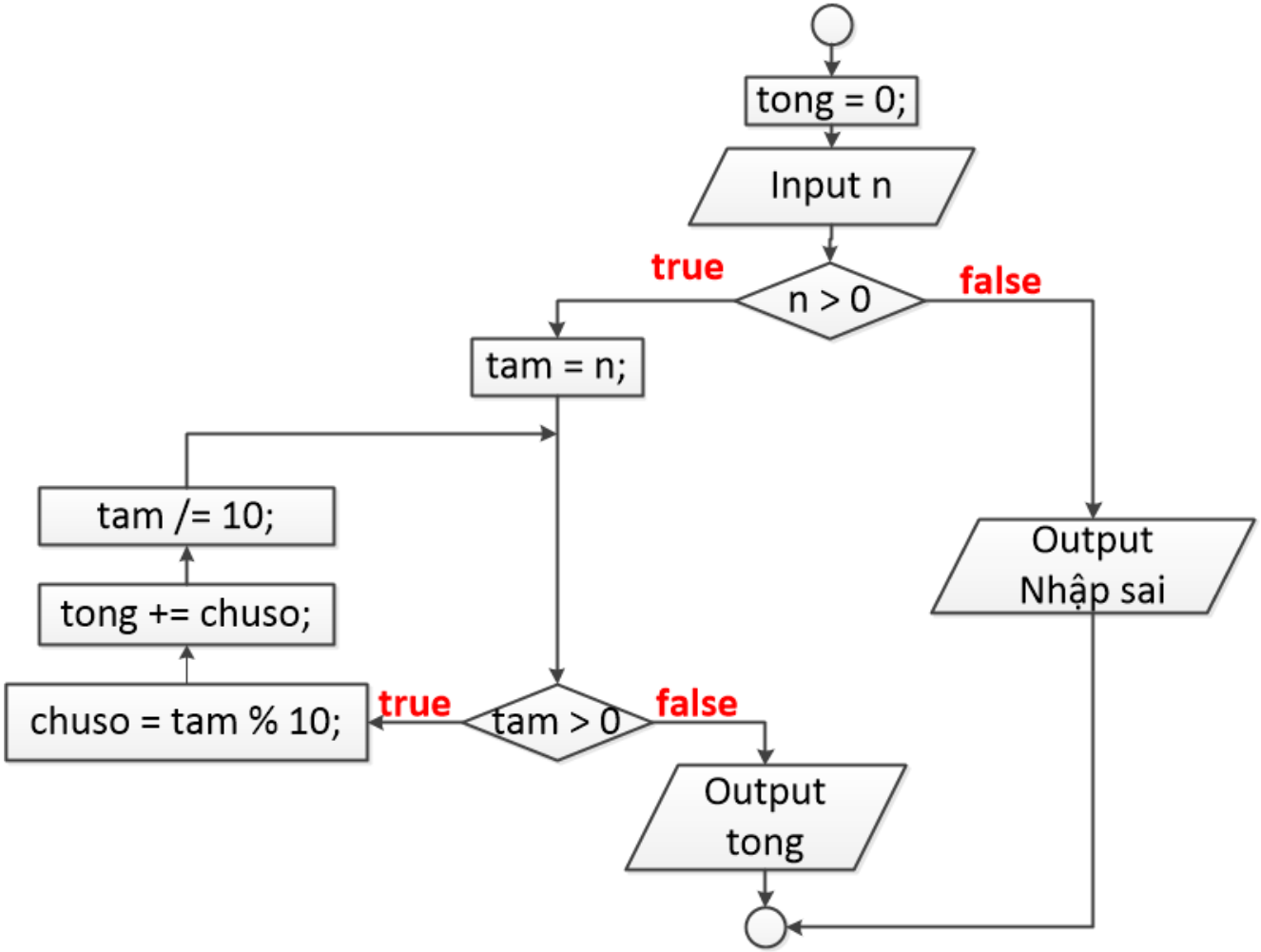
Viết chương trình nhận vào một số nguyên dương n. Tính tổng các chữ số của số nguyên đó.

Input	Processing	Output

Số nguyên dương n	• Nhập số nguyên dương n
	• Xét xem n > 0. Nếu đúng thì tiếp tục bước 3, sai thì xuất nhập sai và kết thúc.
	• Gán n cho tam; khởi tạo tong = 0;
	• Lặp lại các bước sau nếu tam > 0:
	  • Tính chuso = tam % 10;
	  • Tính tong = tong + chuso;
	  • Cập nhật tam = tam / 10;
	• Xuất tong.
		Xuất kết quả tổng các chữ số của n

while

- **Ví dụ 4 dùng giá trị cầm canh (`sentinel value`)**



while

- **Ví dụ 4 dùng giá trị cầm canh (`sentinel value`)**

```
#include <iostream>
using namespace std;
int main()
{
   int n, tam, tong = 0;
   cout << "Nhap so nguyen duong: ";
   cin >> n;
```

while

- **Ví dụ 4 dùng giá trị cầm canh (`sentinel value`)**

```
if (n > 0)
{
  tam = n;
  while (tam > 0)
  {
    int chuso = tam % 10;
    tong += chuso;
    tam /= 10;
  }
  cout << "Tong cac chu so cua " << n << " la "
  << tong << endl;
}
else
  cout << "Nhap sai\n";
return 0;
}
```

while

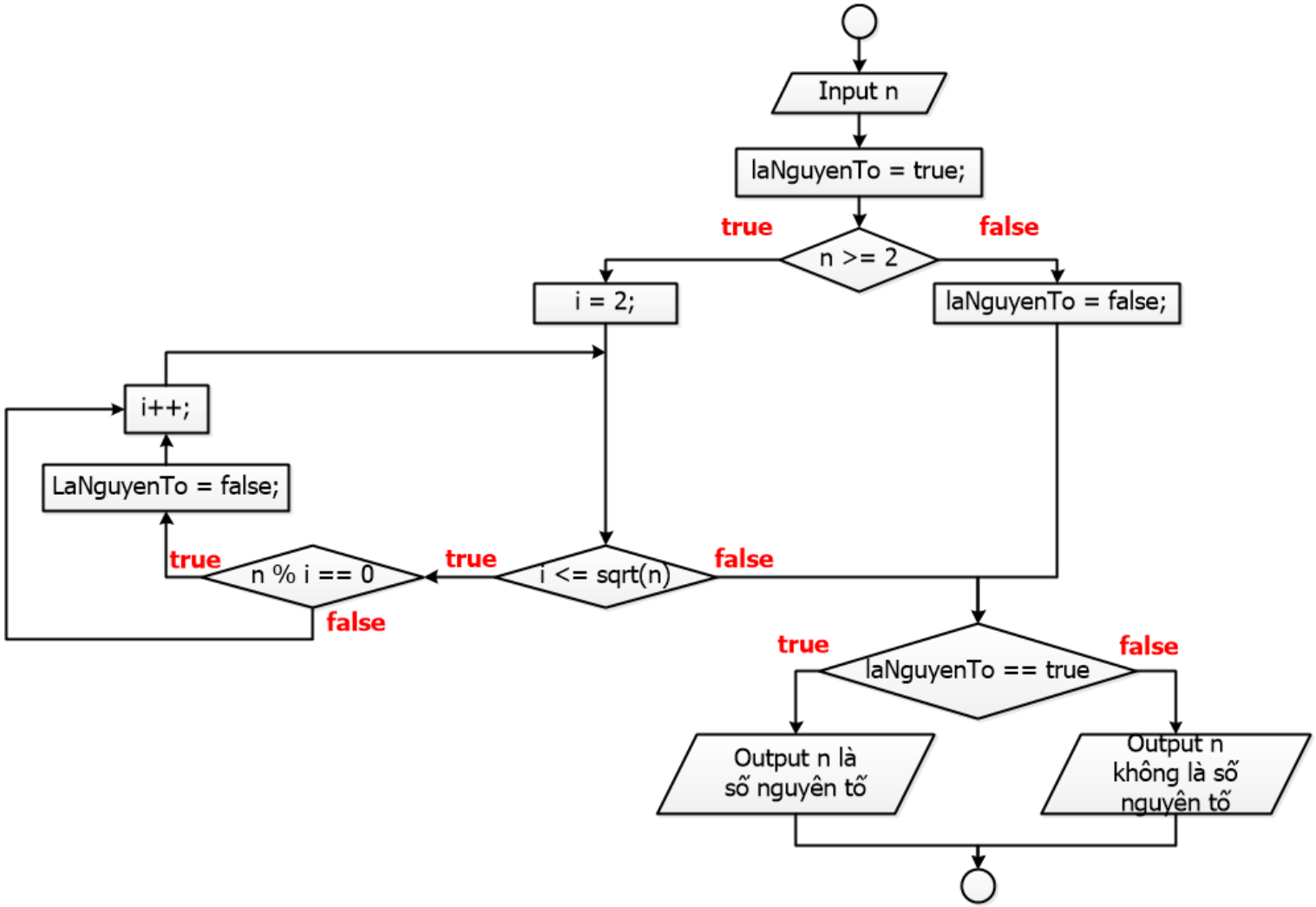
- **Ví dụ 5 dùng biến cờ (`flag`)**

Viết chương trình kiểm tra một số nguyên n được nhập vào có phải là số nguyên tố hay không? Biết rằng số nguyên tố là số từ 2 trở lên, chỉ chia hết cho 1 và cho chính nó.

- **Input**: số nguyên n
- **Output**: n là số nguyên tố/ n không là số nguyên tố.

while

- **Ví dụ 5 dùng biến cờ (`flag`)**



while

- **Ví dụ 5 dùng biến cờ (flag)**

```
#include <iostream>
#include <cmath>
using namespace std;
int main()
{
   int n;
   bool laNguyenTo = true;
   cout << "Nhap so nguyen: ";
   cin >> n;
```

while

- **Ví dụ 5 dùng biến cờ (`flag`)**

```
if ( n >= 2)
{
   int i = 2;
   while (i <= sqrt (static_cast<double>(n)))
   {
      if ( n % i == 0 )
         laNguyenTo = false;
      i++;
   }
}
else
   laNguyenTo = false;
if (laNguyenTo == true)
   cout << n << " la so nguyen to\n";
else
   cout << n << " khong la so nguyen to\n";
return 0;
}
```

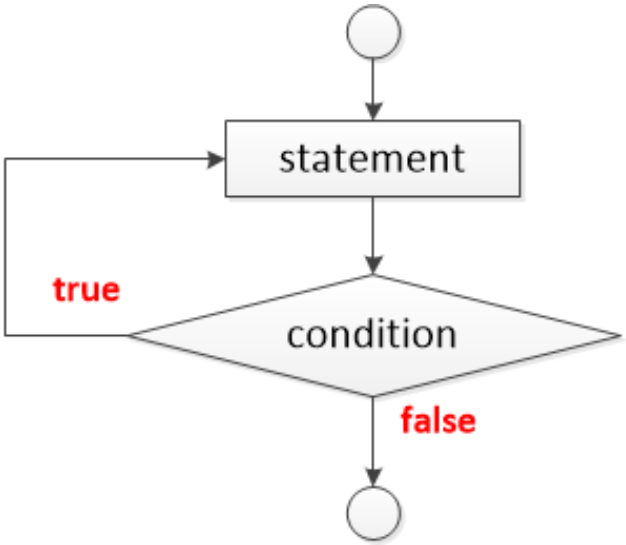
Cấu trúc lặp

- Lệnh while
- *Lệnh do-while*
- Lưu ý khối lệnh trong while và do-while
- Lệnh for
- Một số lưu ý

do-while

- **Cú pháp**

```
do{
    statement;
} while (condition);
```



- **condition** (điều kiện): là biểu thức luận lý, có giá trị true/false.
- **statement** (câu lệnh): có thể là lệnh rỗng, lệnh đơn hay khối lệnh.
- do-while sẽ thực hiện câu lệnh trước, sau đó kiểm tra điều kiện. Thực hiện cho đến khi điều kiện sai thì dừng.

do-while

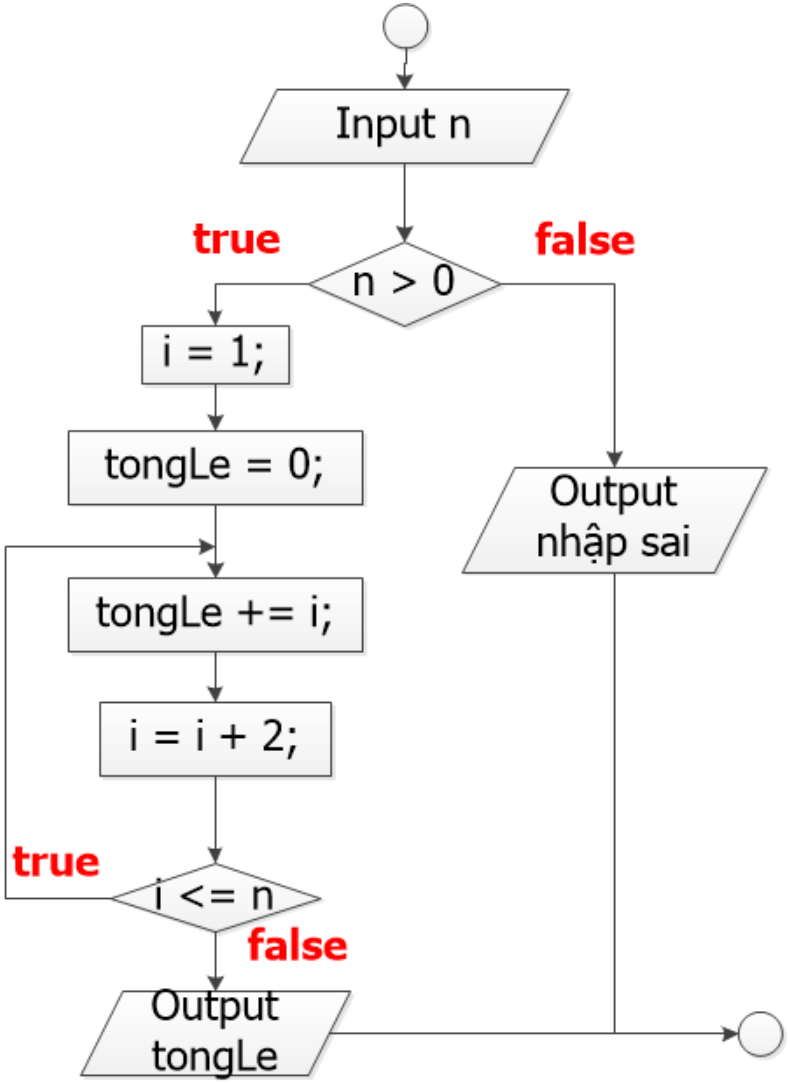
- **Ví dụ 1**

Viết chương trình nhập số nguyên dương n. Tính tổng các số lẻ từ 1 đến n.

- **Input**: số nguyên dương n
- **Output**: tổng các số lẻ từ 1 đến n

do-while

- **Ví dụ 1**



do-while

- **Ví dụ 1**

```
#include <iostream>
using namespace std;
int main()
{
    int n;
    int i = 1;
    int tongLe = 0;
    cout << "Nhap so nguyen duong: ";
    cin >> n;
    if ( n > 0 )
    {
        do{
            tongLe += i;
            i = i + 2;
        } while ( i <= n);
        cout << "Tong cac so le tu 1 den " << n << " la " <<
        tongLe << endl;
    }
    else
        cout << "Nhap sai\n";
    return 0;
}
```

Lưu ý khối lệnh trong while và do-while

- Các lệnh (**statements**) trong **while** và **do-while** thường là **khối lệnh** (từ 2 lệnh trở lên).
- Khối lệnh của **while, do-while** phải chứa câu lệnh **cập nhật** giá trị biến đếm/ biến cầm canh/ giá trị xét điều kiện.

Cấu trúc lặp

- Lệnh while
- Lệnh do-while
- *Lệnh for*
- Một số lưu ý

for

- **Cú pháp**

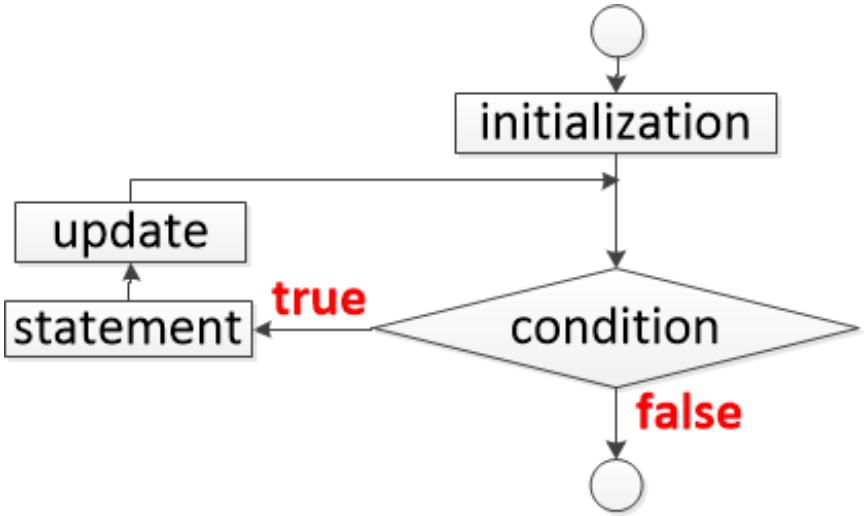
```
for (initialization; condition; update)
    statement;
```

- Trong đó:
  - **initialization:**khởi tạo cho biến đếm.
  - **condition** (điều kiện lặp): là biểu thức luận lý, có giá trị **true/false**.
  - **update**: biểu thức cập nhật giá trị của biến đếm.
  - **statement** (câu lệnh): có thể là lệnh rỗng, lệnh đơn hay khối lệnh. Câu lệnh sẽ được thực hiện nếu **condition** có giá trị **true**.

for

- **Cú pháp**

```
for (initialization; condition; update)
    statement;
```

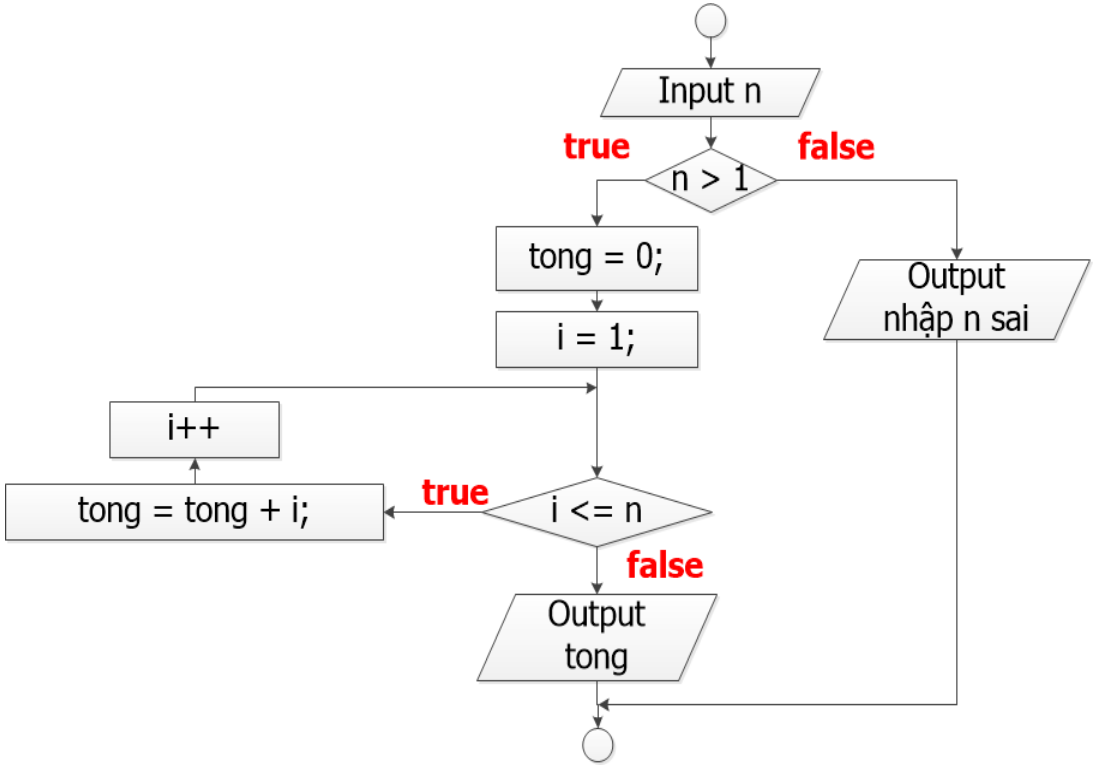


for

- **Ví dụ 1**

Viết chương trình nhập vào một số nguyên dương n.
Tính: S = 1 + 2 + 3 + … + n

Input: số nguyên dương n
Output: tổng từ 1 đến n



for

- **Ví dụ 1**

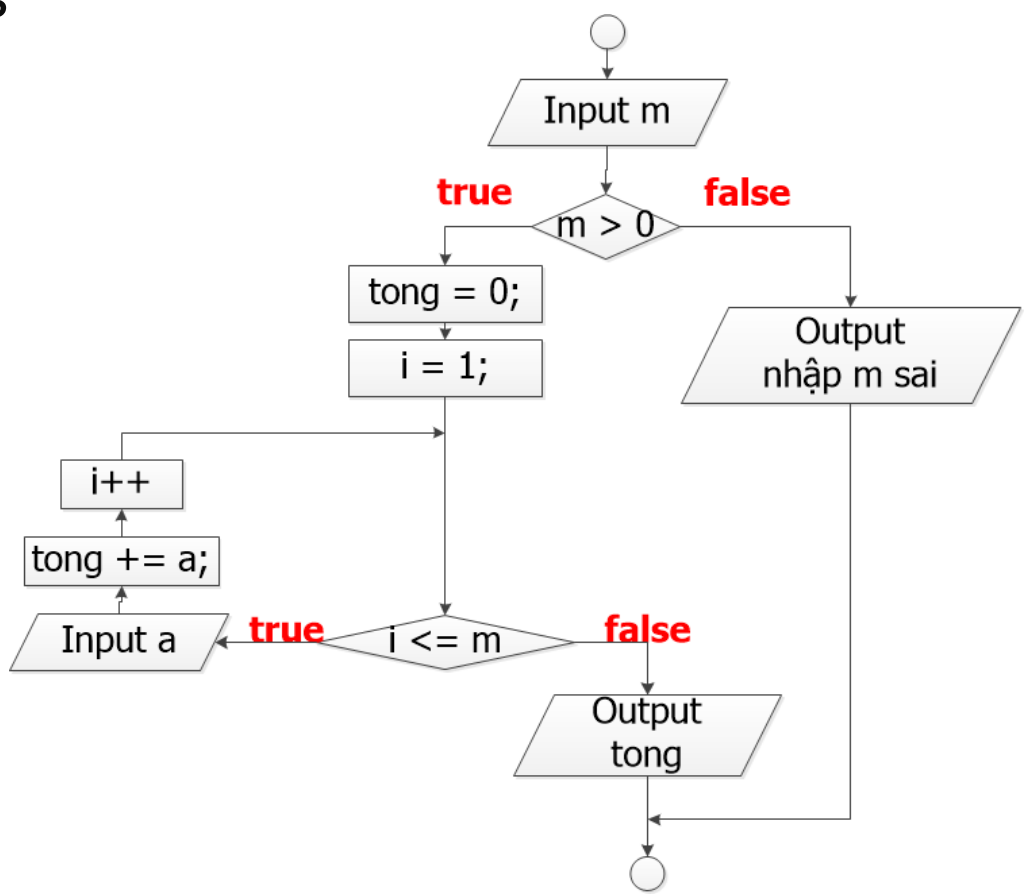
```
#include <iostream>
using namespace std;
int main()
{
    int n;
    int tong = 0;
    cout << "Nhap so nguyen duong n: ";
    cin >> n;
    if (n > 0)
    {
        for (int i = 1; i <= n; i++)
            tong += i;
        cout << "Tong cac so tu 1 den " << n << " la " <<
        tong << endl;
    }
    else
        cout << "Nhap n sai\n";
    return 0;
}
```

for

- **Ví dụ 2**

Viết chương trình tính tổng m số nguyên dương nhập vào.

- **Input**: m số nguyên dương
- **Output**: tổng m số nguyên dương đã nhập



for

- **Ví dụ 2**

```
#include <iostream>
using namespace std;
int main()
{
    int m, a;
    int tong = 0;
    cout << "Nhap so luong so: ";
    cin >> m;
    if (m > 0)
    {
        for (int i = 1; i <= m; i++)
        {
            cout << "Nhap so thu " << i << ": ";
            cin >> a;
            tong += a;
        }
        cout << "Tong " << m << " so vua nhap la " << tong << endl;
    }
    else
        cout << "Nhap sai\n";
    return 0;
}
```

Cấu trúc lặp

- Lệnh while
- Lệnh do-while
- Lệnh for
- *Một số lưu ý*

Một số lưu ý - cấu trúc lặp

- **`while`** và **`do-while`**e thường sử dụng khi không biết trước số lần lặp; **`for`** thường sử dụng khi biết trước số lần lặp.
- Tất cả lệnh lặp đều có khả năng lặp vô tận. **`for`**, **`while`** có thể không xảy ra lần lặp nào; **`do`**…**`while`** ít nhất 1 lần lặp.
- Không nên thay đổi giá trị biến đếm bên trong câu lệnh ở thân vòng lặp **`for`** (vì đã có biểu thức cập nhật biến đếm).

Một số lưu ý - cấu trúc lặp

- Lệnh **do-while** thường dùng để kiểm tra dữ liệu hợp lệ, cho phép nhập lại nếu sai.

Ví dụ: yêu cầu người dùng nhập điểm có giá trị là số nguyên từ 0 đến 10.

```
int diem;
do
{
      cout << "Nhap diem tu 0 den 10: ";
      cin >> diem;
      if ( diem < 0 || diem > 10 )
         cout << "Nhap diem sai. Nhap lai\n";
} while (diem < 0 || diem > 10);
```

Lệnh break và continue

- Lệnh break
- Lệnh continue

Lệnh break

- Câu lệnh **`break;`** dùng để bỏ qua phần còn lại trong câu lệnh **`switch`**.
- Câu lệnh **`break;`** khi được thực hiện bên trong vòng lặp **`for`**, **`while`**, **`do-while`** sẽ thoát khỏi vòng lặp.
- Ví dụ:

```
int tong = 0;
for (int i = 1; i <= 5; i++)
{
        if (i == 3)
            break;
        tong += i;
}
cout << "Tong la: " << tong << endl;
//tong la 3
```

Lệnh continue

- Câu lệnh **`continue;`** trong vòng lặp **`for`**, **`while`**, **`do-while`** dùng để bỏ qua phần còn lại trong lần lặp đó và bắt đầu lần lặp kế tiếp.
- Ví dụ:

```
int tong = 0;
for (int i = 1; i <= 5; i++)
{
   if (i == 3)
         continue;
   tong += i;
}
cout << "Tong la: " << tong << endl;
//tong la 12
```

Q & A