

# BT ôn tập: Cho biết kết quả a, b, c đoạn lệnh sau:

```
int a=5, b=10, c=7;
```

```
if (a>b)
```

```
    a = 0;
```

```
if (b>c)
```

```
{
```

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

```
    c = a+c;
```

```
}
```

```
if (c>a)
```

```
{c = c-a;
```

```
  b= c+a;}
```

```
int a=3, b = 7, c=1;
```

```
if (a>c)
```

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

```
    b = b+2;
```

```
    c = b - a;
```

```
int a=3, b = 7, c=1;
```

```
if (a<c){
```

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

```
    b = b+2;}
```

```
c = b - a;
```

# Chương 3. CT cấu trúc trong C

1. Kỹ thuật giải quyết vấn đề cơ bản
2. Phát triển thuật toán
3. Sử dụng **if** và **if ..... else**
4. Sử dụng lệnh **while**
5. Vòng lặp điều khiển đếm và giá trị canh
6. Cấu trúc điều khiển lồng nhau
7. Phép gán
8. Phép tăng giảm

# Giới thiệu

- Trước khi viết CT
  - Hiểu yêu cầu
  - Lên kế hoạch giải quyết
- Trong khi viết chương trình
  - Lời giải sẵn cho các khối nhỏ
  - Sử dụng các nguyên tắc lập trình

# 1. Thuật toán và Pseudocode

- Thuật toán: Là các hành động được thực hiện theo một trật tự nào đó
  - Tất cả vấn đề tính toán đều được giải quyết bằng việc thực thi các hoạt động có trật tự nào đó
  - Điều khiển chương trình: xác định thứ tự thực hiện lệnh
- Pseudocode (mã giả): Là ngôn ngữ không chính thức để viết thuật toán
  - Tương tự ngôn ngữ tiếng Anh hàng ngày
  - Không thực thi được trên máy tính
  - Giúp suy xét chương trình trước khi viết
  - Đặc điểm: Dễ chuyển sang C; Chỉ gồm các lệnh (không khai báo biến)

# VD:

VD 1: Tìm số lớn hơn trong 2 số

- Enter 2 integer numbers: x,y
- If  $(x > y) \rightarrow$  Print x
- Else print y

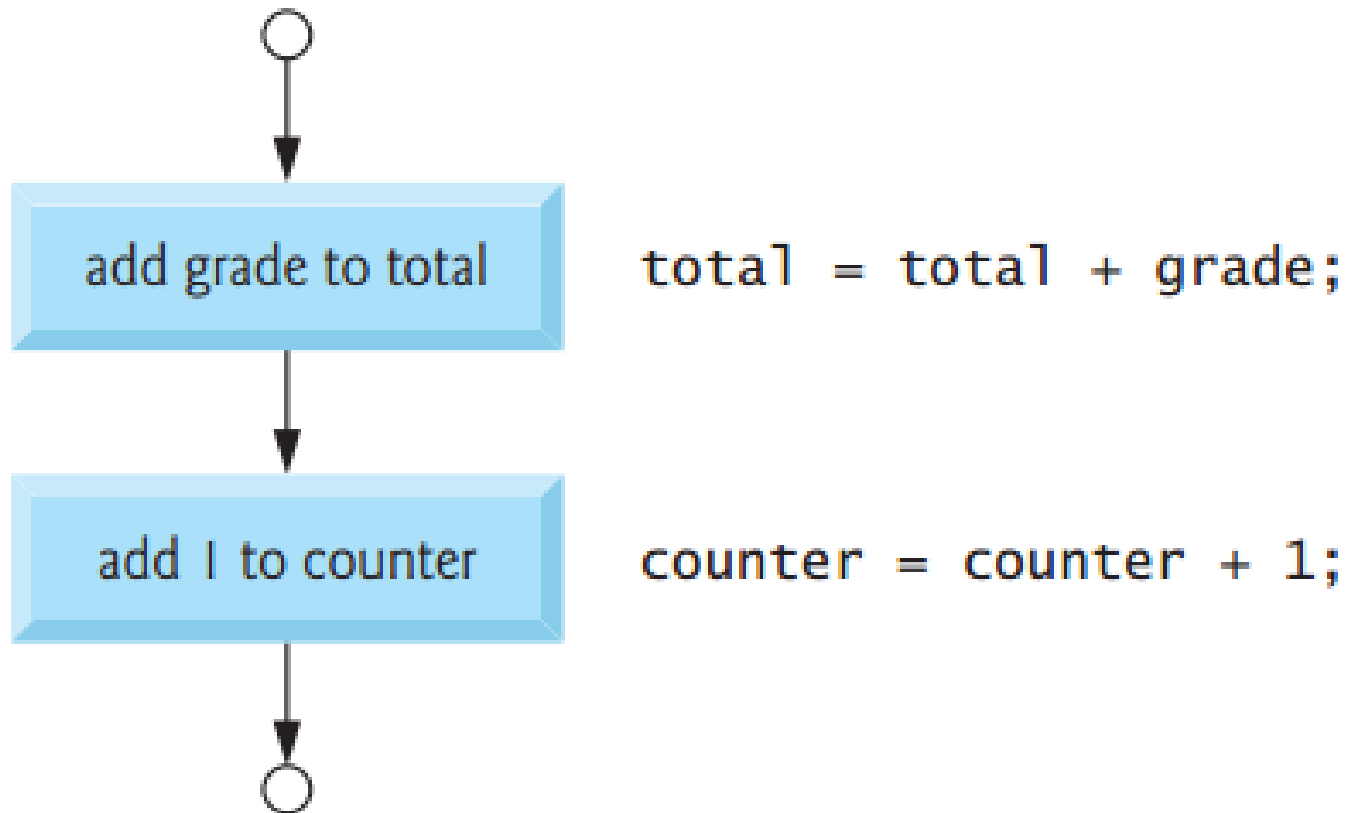
VD 2: Giải phương trình bậc hai

- Enter a,b,c
- If  $a \neq 0$ 
  - Count Delta
  - If  $\Delta > 0 \rightarrow$  Print: pt có 2 nghiệm phân biệt x1, x2
  - If  $\Delta = 0 \rightarrow$  Print: pt có nghiệm kép:  $x_1 = x_2 = \dots$
  - Else  $\Delta < 0 \rightarrow$  Print: pt vô nghiệm
- Else
  - print: không phải pt bậc 2

## 2. Các cấu trúc điều khiển

- Thực hiện tuần tự: Các lệnh được thực hiện tuần tự
- Chuyển điều khiển:
  - Khi lệnh tiếp theo được thực hiện không là lệnh tiếp theo trong chuỗi lệnh
  - Lạm dụng lệnh goto sẽ gây ra nhiều vấn đề
- Bohm & Jacobini: Tất cả các CT đều được viết với 3 cấu trúc điều khiển
  - Tuần tự
  - Cấu trúc lựa chọn: if; if ... else; switch
  - Cấu trúc vòng lặp: while, do ... while, for

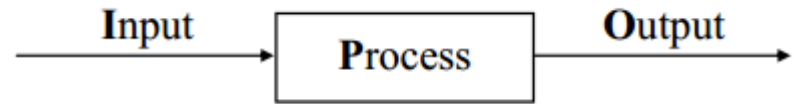
# Cấu trúc tuần tự



# Các bước lập trình

- Bước 1: Phân tích vấn đề và xác định các đặc điểm. (xác định I-P-O)

- I-P-O:



- VD: Xác định I-P-O của tính tiền công nhật  
Giải pt bậc nhất









- Bước 2: Lập ra giải pháp. (đưa ra thuật giải)
- Bước 3: Viết chương trình
- Bước 4: Chạy thử chương trình
- Bước 5: Kiểm chứng và hoàn thiện CT



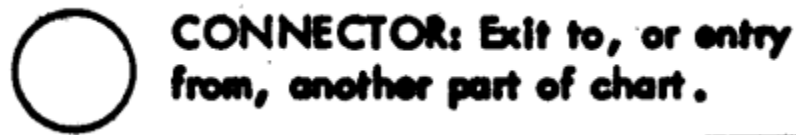
## 2. Lưu đồ

- Là biểu diễn đồ họa của 1 thuật toán hoặc 1 phần của thuật toán
- VD1:
- Sử dụng các hình đặc biệt như sau để vẽ lưu đồ

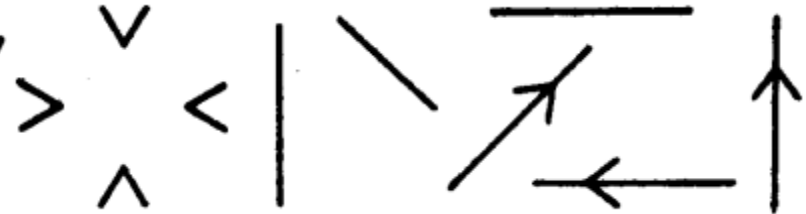
## 2. Lưu đồ

Hình dạng (symbol)	Hành động (Activity)
	Dữ liệu vào (Input)
	Xử lý (Process)
	Dữ liệu ra (Output)
	Quyết định (Decision), sử dụng điều kiện
	Luồng xử lý (Flow lines)
	Gọi CT con, hàm... (Procedure, Function...)
	Bắt đầu, kết thúc (Begin, End)
	Điểm ghép nối (Connector)

## 2. Lưu đồ

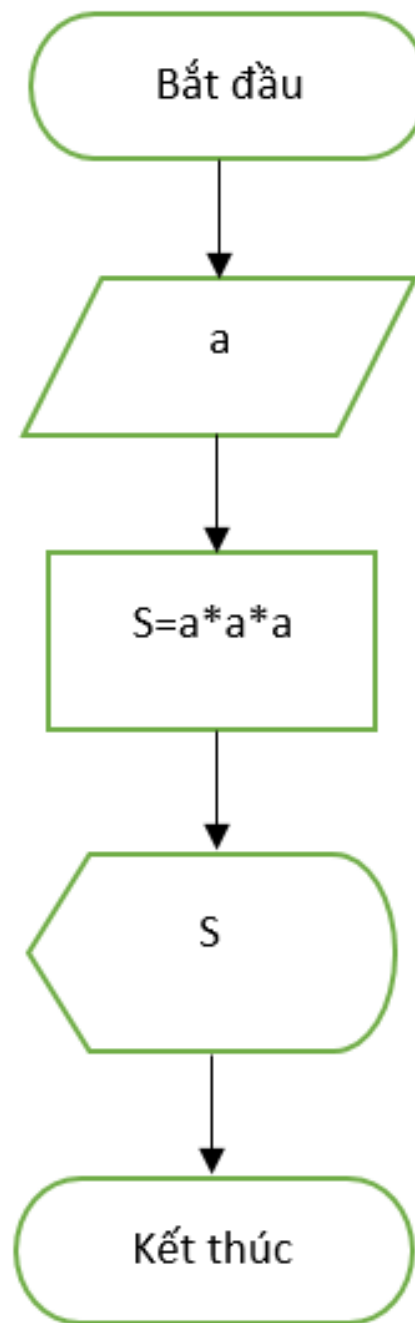


**ARROWHEADS and Flowlines:** In linking symbols, these show operations sequence and dataflow direction. Arrowheads required if path on any linkage is not left-to-right or top-to-bottom.

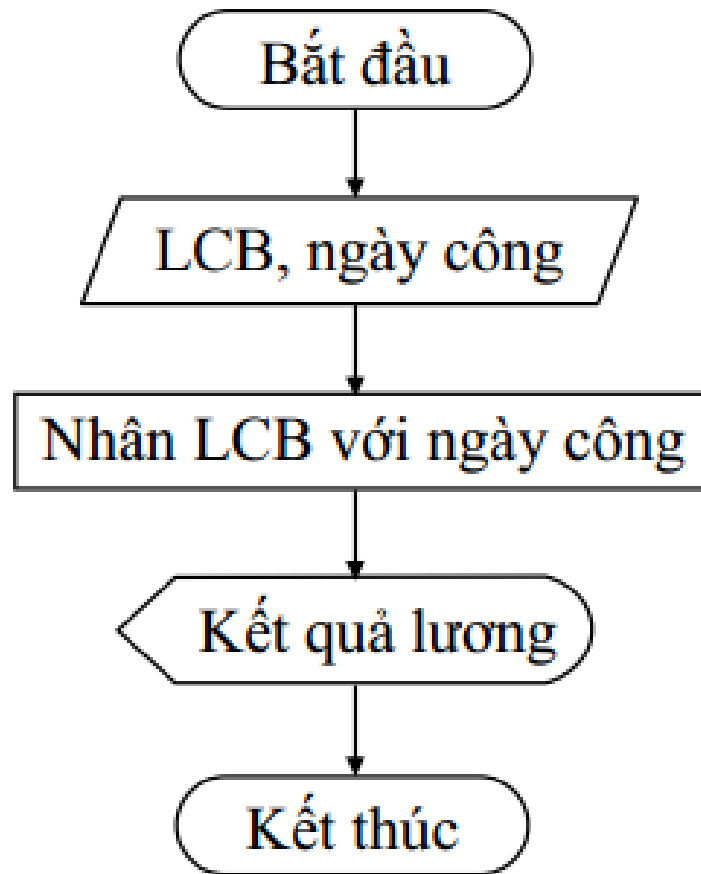


## 2. Lưu đồ

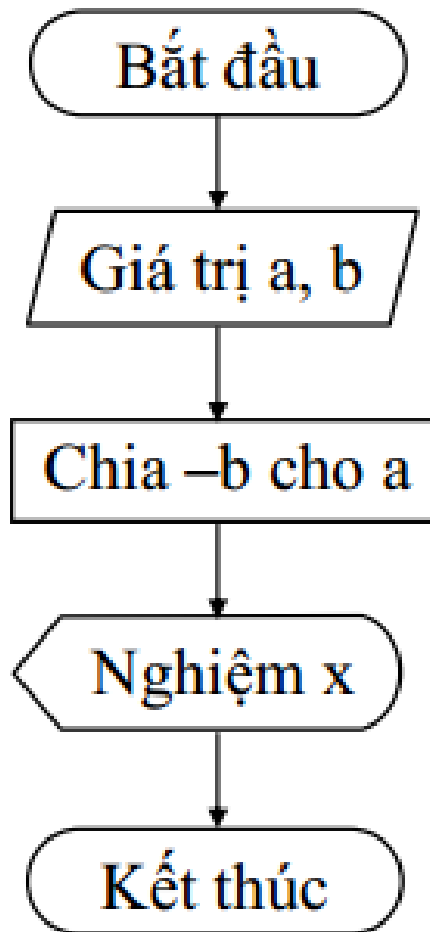
- VD1: Vẽ lưu đồ tính  $S=a*a*a$



## VD2: Vẽ lưu đồ tính tiền công nhật



## VD3: Vẽ lưu đồ giải phương trình bậc nhất

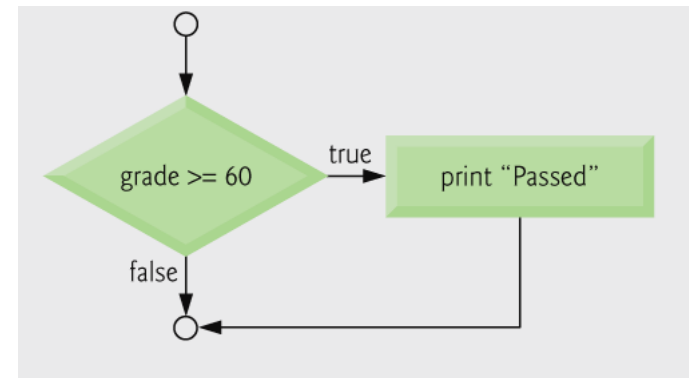


# Bài tập: Vẽ lưu đồ

- Tính điểm trung bình 3 môn Toán, Lý, Hóa
- Kiểm tra số nguyên chẵn hay lẻ
- Báo đậu rớt cho SV (điểm  $\geq 5$ : đậu)
- Giải phương trình bậc hai

# 4. Cấu trúc if

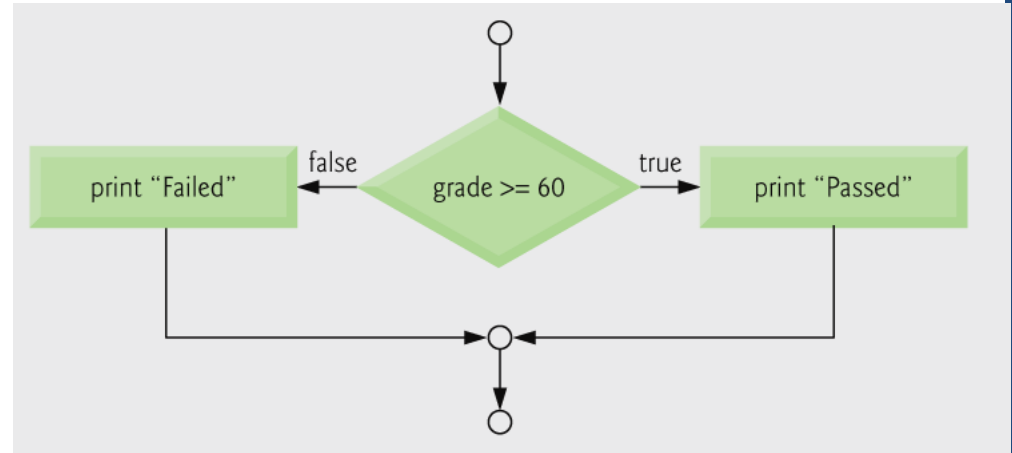
- Lựa chọn trong các điều kiện của một hành động
- Chỉ thực hiện nếu điều kiện đúng
- VD: In ra kết quả SV đậu (đậu nếu điểm  $\geq 60$ )
  - Nếu if đúng thì thực hiện in “passed”
  - Nếu if sai bỏ qua lệnh in và tiếp tục thực hiện lệnh khác (nếu có)
  - Ngôn ngữ tự nhiên:  
    nếu (grade $\geq 60$ )  
        in (“passed”);
  - Lưu đồ:
  - Câu lệnh:  
    if (grade $\geq 60$ )  
        printf (“passed\n”);





# 4.1. Cấu trúc if....else

- Mô tả hoạt động khi điều kiện đúng và sai
- VD: In ra kết quả SV đậu và rớt (đậu nếu điểm  $\geq 60$ , ngược lại là rớt)
  - Ngôn ngữ tự nhiên
  - Lưu đồ
  - Lệnh
- Có 3 điều cần chú ý:
  - Điều kiện
  - Giá trị nếu if đúng
  - Giá trị nếu if sai



# VD: Vẽ lưu đồ và viết CT

- Tìm và in ra số lớn hơn trong 2 số nguyên
- Tìm và in ra số lớn nhất trong 3 số nguyên

## 4.2. Toán tử điều kiện Ternary

- Cấu trúc:
  - (biểu thức điều kiện) ? (giá trị trả lại nếu true) : (giá trị trả lại nếu false)
- VD1:
  - (Grade >=60) ? Printf (“passed\n”) : printf (“failed\n”);
  - Printf (“%s \n”, grade>=60? “passed” : “failed”);

## 4.3. Cấu trúc if...else lồng

- Kiểm tra nhiều trường hợp bằng cách đặt nhiều lệnh if ... else bên trong if ...else
- Một điều kiện thỏa thì các điều kiện khác bỏ qua
- Nếu điểm  $\geq 90 \rightarrow$  “A”

    Nếu điểm  $\geq 80 \rightarrow$  “B”

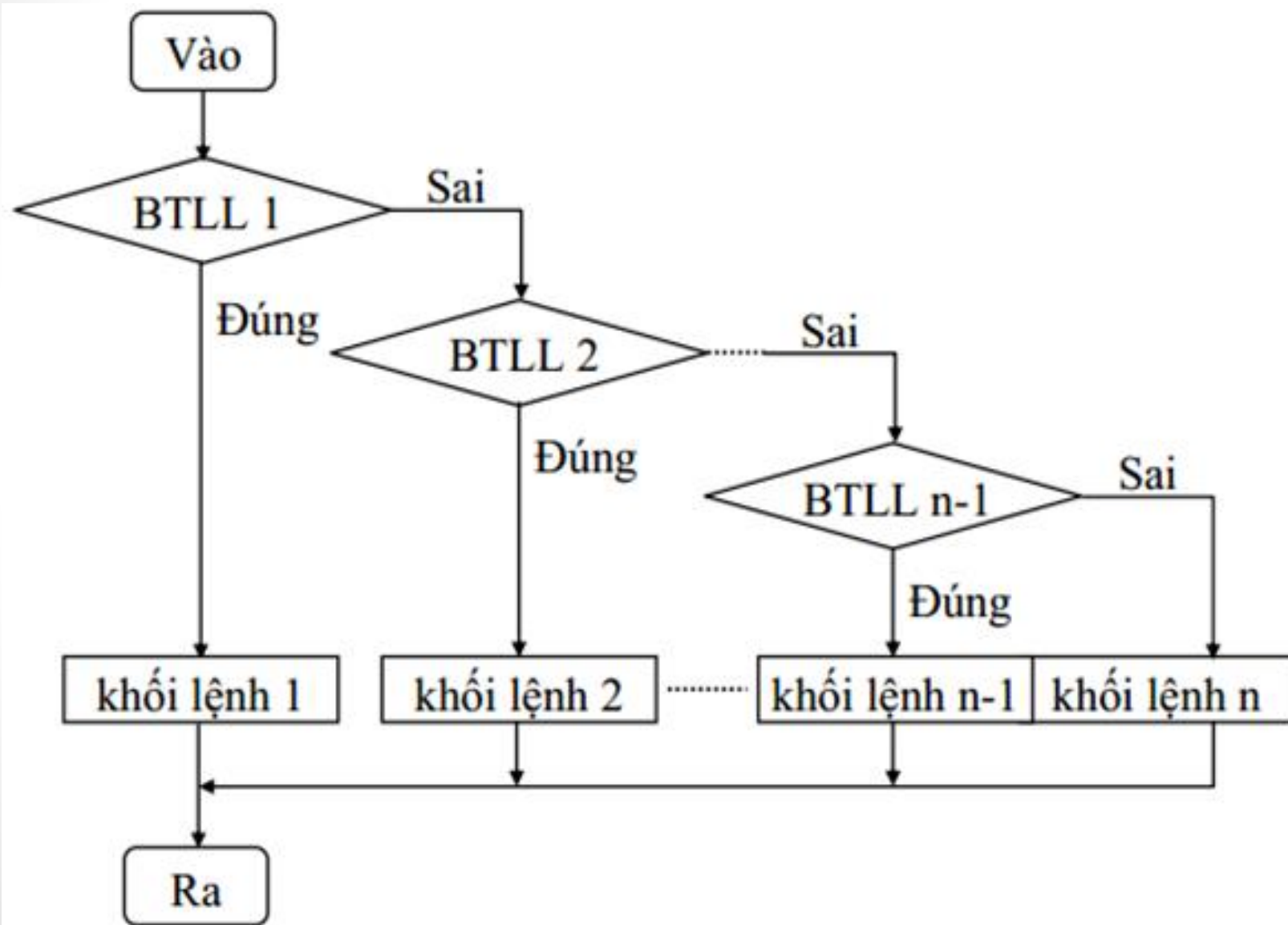
        Nếu điểm  $\geq 70 \rightarrow$  “C”

            Nếu điểm  $\geq 60 \rightarrow$  “D”

                Còn lại  $\rightarrow$  “F”

- *If student's grade is greater than or equal to 90  
Print "A"*  
*else*  
*If student's grade is greater than or equal to 80  
Print "B"*  
*else*  
*If student's grade is greater than or equal to 70  
Print "C"*  
*else*  
*If student's grade is greater than or equal to 60  
Print "D"*  
*else*  
*Print "F"*

## 4.3. Cấu trúc if...else lồng



```
#include <stdio.h>
#include <conio.h>
void main(void){
int grade;
printf("enter grade: \n");
scanf("%d", &grade);
if (grade >= 90) {
    puts("A");}
else {
    if (grade >= 80) {
        puts("B");}
    else {
        if (grade >= 70) {
            puts("C");}
        else {
            if (grade >= 60) {
                puts("D");}
            else {
                puts("F");}
            } // end else
        } // end else
    } // end else
getch();
}
```

# Lệnh Logic

- **&&** Lệnh AND
  - Trả về TRUE nếu cả 2 điều kiện TRUE
- **||** Lệnh OR
  - TRẢ về TRUE nếu một trong hai điều kiện TRUE
- **!** Lệnh NOT
- BTT

Biểu thức	Kết quả
True && false	False
True    false	True
! false	True



# Các lỗi

- Syntax error
  - Lỗi cấu trúc được phát hiện bởi trình biên dịch
- Logic errors
  - Lỗi logic xuất hiện trong thời gian thực hiện (Khó phát hiện)
  - Lỗi logic nghiêm trọng (fatal error): Dừng CT
  - Lỗi logic non-fatal không làm dừng chương trình nhưng sai kết quả

# Exercise 1

- Cho biết giá trị của các biến a và b sau khi thực hiện chương trình

```
int a = 0, b = 5, c = 3;
```

```
if ((a > b) || (b < c))
```

```
    a = b + c;
```

```
else
```

```
    b = a + c;
```

# Exercise 2

- Cho biết giá trị của các biến a, b, và c sau khi thực hiện chương trình

```
int a = 5, b = 6, c = 7;  
if ( a > c)  
    a = a + b;  
b = b + 1;  
if ( a < b)  
{  
    c = b - a;  
    b = c + 5;  
}
```

Kết quả : a = ..... b = ..... c =.....

# Exercise 3

- Cho biết giá trị của các biến a và b sau khi thực hiện chương trình

```
int a = 4, b = 2, c = 5;
```

```
c -= 2;
```

```
if ( a > c)
```

```
    a = a + b;
```

```
b = b + c;
```

```
if ( a < b)
```

```
{
```

```
    c = b - a;
```

```
    b = c + 5;
```

```
}
```

```
else
```

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

```
b = b + 1;
```

Kết quả : a = ..... b = ..... c = .....

# Exercise 4

- Cho biết giá trị của các biến a và b sau khi thực hiện chương trình

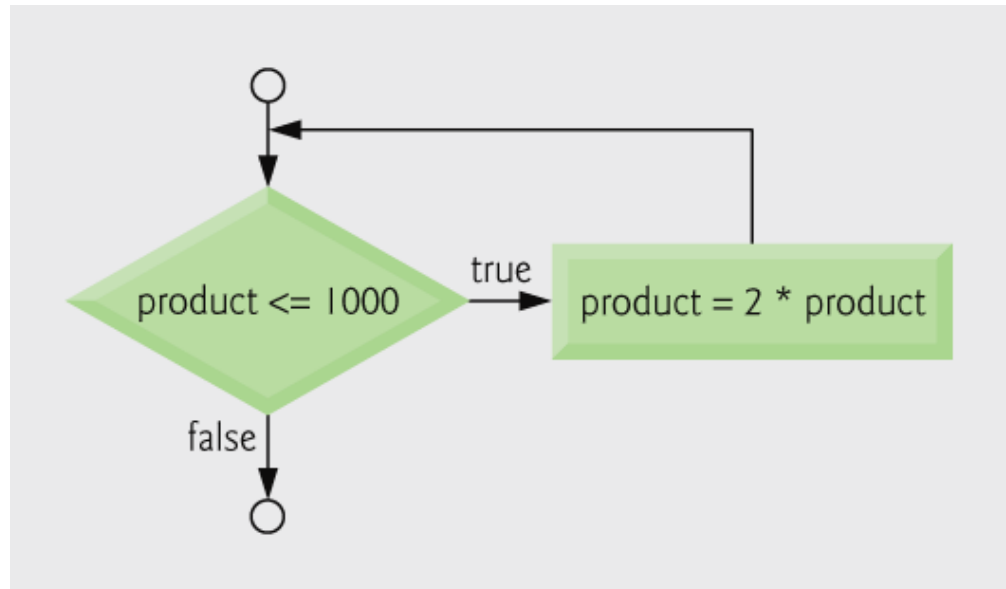
```
int a = 4, b = 2, c = 5;
if ((a > c) || (a < b) && (b > c))
    a = a - 2;
else if (a == b)
{
    c = b - a;
    b = c + 5;
}
else
    a = a + 1;
b = b + 1;
```

Kết quả : a = ..... b = ..... c = .....

# 5. Cấu trúc lặp while

- Lặp lại nhiều lần khi điều kiện vẫn đúng
- Vòng lặp while lặp lại cho đến khi điều kiện sai
- VD: `int product = 2;`

`While (product <=100)`  
`product = 2* product;`



# 5.1. Vòng lặp điều khiển đếm

- Khi vòng lặp lặp lại  $\rightarrow$  Bộ đếm đạt giá trị nhất định
- Vòng lặp xác định: Số vòng lặp biết trước
- VD: Có 10 SV làm kiểm tra, điểm từ 0  $\rightarrow$  100. Tính điểm TB

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
    unsigned int counter;
    int grade, total, average;
    total = 0;
    counter = 1;
    while (counter <= 10)
        { // loop 10 times
            printf("%s", "Enter grade: ");
            scanf("%d", &grade);
            total = total + grade;
            counter = counter + 1;
        }
    average = total / 10;
    printf("Class average is %d\n", average);
    getch();
}
```



## 5.2. Xác định thuật toán sàng lọc trên dưới và từng bước

- Vấn đề: Xác định điểm TB với số SV không biết trước
- Làm sao để biết CT kết thúc? số vòng lặp?
- Sử dụng giá trị canh (Sentinel value) để chỉ kết thúc nhập dữ liệu
- Người sử dụng nhập giá trị này nếu muốn kết thúc (có gợi ý)
- Giá trị này được chọn để không bị nhầm với giá trị nhập thông thường (TH này là -1)
- VD: Tính TB điểm nhập vào, kết thúc nhập khi nhập giá trị -1

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
    int counter; // number of grade to be entered next
    int grade, total;
    float average; // grade value
    total = 0;
    counter = 0;
    printf("%s", "Enter grade, (-1 to end): "); // prompt for
    input
    scanf("%d", &grade); // read next grade
    while (grade != -1)
    {
        total = total + grade; // add grade to total
        counter = counter + 1; // increment counter
        printf("%s", "Enter grade, (-1 to end): ");
        scanf("%d", &grade); }
    if (counter != 0)
    {
        average = (float)total / counter;
        printf("Class average is %.2f \n", average); }
    else
    puts("No grades were entered");
    getch();
}
```

## 5.2. Xác định thuật toán sàng lọc trên dưới và từng bước

- $\text{average} = (\text{float})\text{total} / \text{counter};$
- total và counter là số nguyên, average là số thực
- Thử lại với không sử dụng (float)
- Giải thích kết quả nếu không có dấu { } sau while
- Giải thích kết quả nếu không có “.” trong lệnh printf

## 5.3. Các cấu trúc lồng nhau

- Có 1 list kết quả kiểm tra cho 10 SV (1: đậu, 2 rớt).  
Viết CT phân tích kết quả, nếu  $\geq 8$  SV đậu thì in ra có thưởng
- Truy xuất 10 kết quả kiểm tra  $\rightarrow$  vòng lặp điều khiển đếm

Đếm đậu và rớt  $\rightarrow$  2 bộ đếm

Mỗi kết quả kiểm tra là 1 con số (1 hoặc 2)

## 5.3. Các cấu trúc lồng nhau

### Pseudocode

- Student<=10
  - {Enter next result
    - if result=1
      - passes = passes+1
    - else
      - failures = failures+1
  - student+1}
  - print (passes, failures)
  - if passes>=8
    - print (bonus)

# 6. Phép gán và phép tăng giảm

- Viết tắt phép gán:

Biến	=	Biến	toán tử	biểu thức
Biến		toán tử	=	biểu thức

- VD: 
$$\begin{array}{ccccccc} c & = & c & + & & 3 \\ c & & + & & = & 3 \end{array}$$

- VD: 
$$\begin{aligned} c+=7 &\rightarrow c = c + 7 \\ d-=4 &\rightarrow d = d - 4 \\ e*=5 &\rightarrow e = e*5 \\ f/=3 &\rightarrow f = f/3 \\ g\%=9 &\rightarrow g = g\%9 \end{aligned}$$

# 6. Phép gán và phép tăng giảm

- Phép tăng giảm

**++**  $\rightarrow c = c + 1$  hay  $c+=1$

**--**  $\rightarrow c = c - 1$  hay  $c-=1$

**++c** ; **--c**: Cộng (trừ) trước: Biến thay đổi trước khi biểu thức được thực hiện

**c++** ; **c--** : Cộng (trừ) sau: Biến thay đổi sau khi biểu thức được thực hiện

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

```
int main(void)
{
    int c;
    // demonstrate postincrement
    c = 5; // assign 5 to c
    printf("%d\n", c);
    printf("%d\n", c++);
    printf("%d\n\n", c);
    c = 5; // assign 5 to c
    // demonstrate preincrement
    printf("%d\n", c);
    printf("%d\n", ++c);
    printf("%d\n", c);
    getch();
}
```



```
#include <stdio.h>
#include <conio.h>
void main(void)
{
    int a = 1, b = 5, c = 8, d;
    d = (b++) - a + (++c);
    printf("d = %d", d);
    getch();
}
```

```
#include <stdio.h>
#include <conio.h>
void main(void)
{
    int product = 1;
    while (++product <= 10)
    {
        printf("product:%d\n", product);
    }
    getch();
}
```