First Year Engineer in computer science

CHAPTER 3 :

Conditional Statements

2023-2024

Sid Ahmed BERRABAH

Conditional Statements

Definition:

Conditional statements (also known by selection statement or Decision Making Statements) control the sequence of statement execution, depending on the value of a a **controlling expression**(condition)

C-language supports two conditional statements.

1. if
2. switch.

if Statement

- The if statement controls conditional branching.
- The if Statement may be implemented in different forms.

  1. simple if statement.
  2. if –else statement
  3. nested if-else statement.
  4. if .. else if … else statement.

Simple if statement

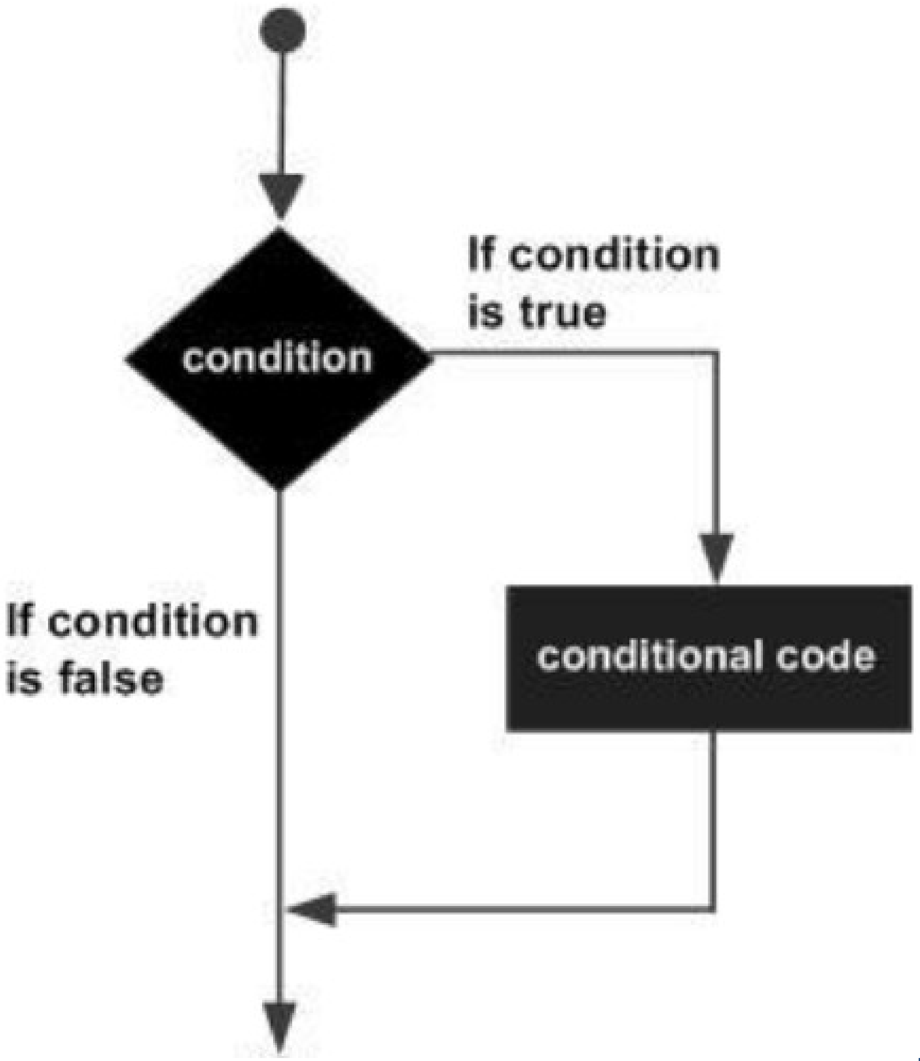
A simple if statement consists of a boolean expression followed by one or more statements.

Algorithm (pseudo-code) :

```
if (boolean_expression) then:
        do_something
end if
```

Simple if statement

Algorithm : Flowchart:



Simple if statement

“C” Syntax:

```
if (boolean_expression)
{
   /* statement(s) will execute if the boolean expression is true */
}
```

- If the boolean expression evaluates to **true**, then the block of code inside the if statement will be executed. If boolean expression evaluates to **false**, then the first set of code after the end of the if statement (after the closing curly brace) will be executed.

- C programming language assumes any **non-null** values as **true** and if it is **null** then it is assumed as **false** value.

- In if statement, a single statement can be included without enclosing it into curly braces { }.

Simple if statement

Example 1: Verify if the variable a is greater than the variable b.

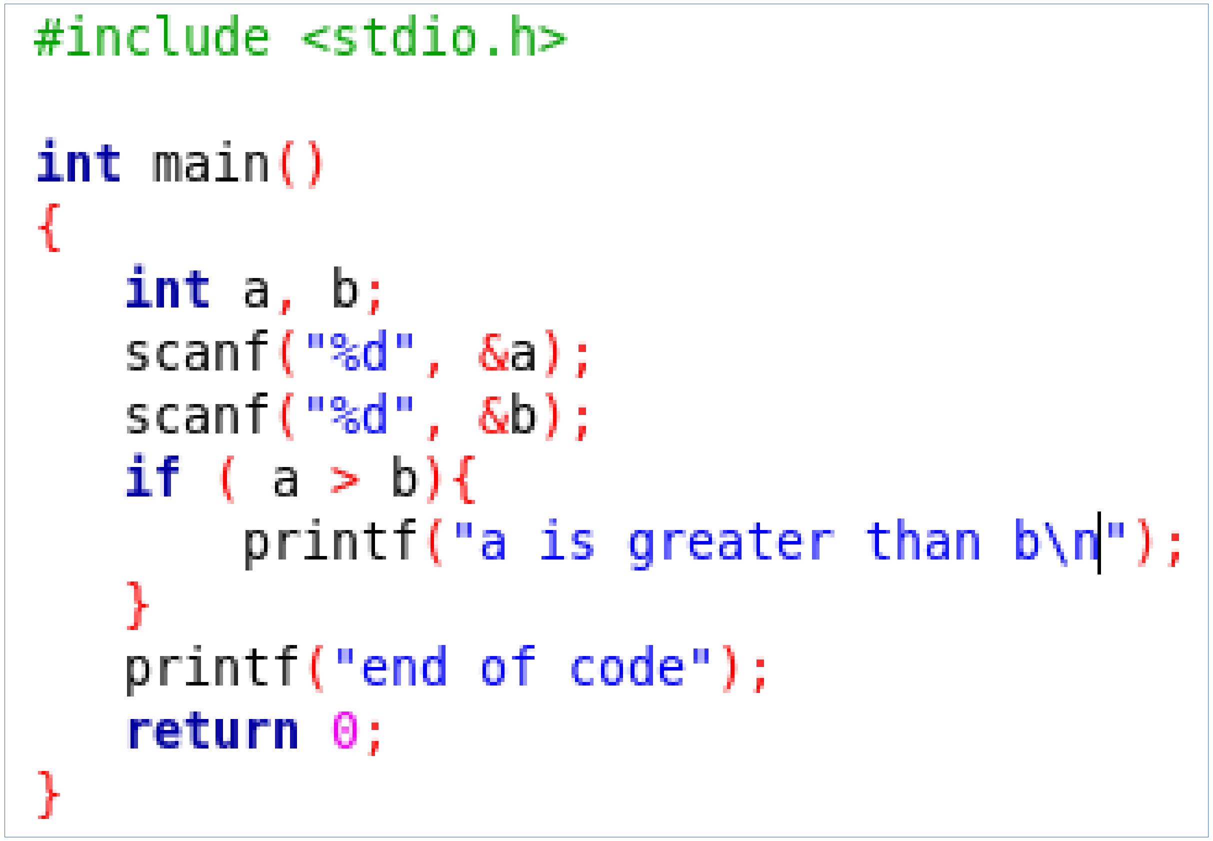
Algorithm:

```
begin
read(a)
read(b)
if (a>b) then
    write (‘a is greater than b’)
endif
end
```

Instruction if …

Example 1: Verify if the variable a is greater than the variable b.

C Program:



```c
#include <stdio.h>

int main()
{
    int a, b;
    scanf("%d", &a);
    scanf("%d", &b);
    if ( a > b){
        printf("a is greater than b\n");
    }
    printf("end of code");
    return 0;
}
```

if...else statement

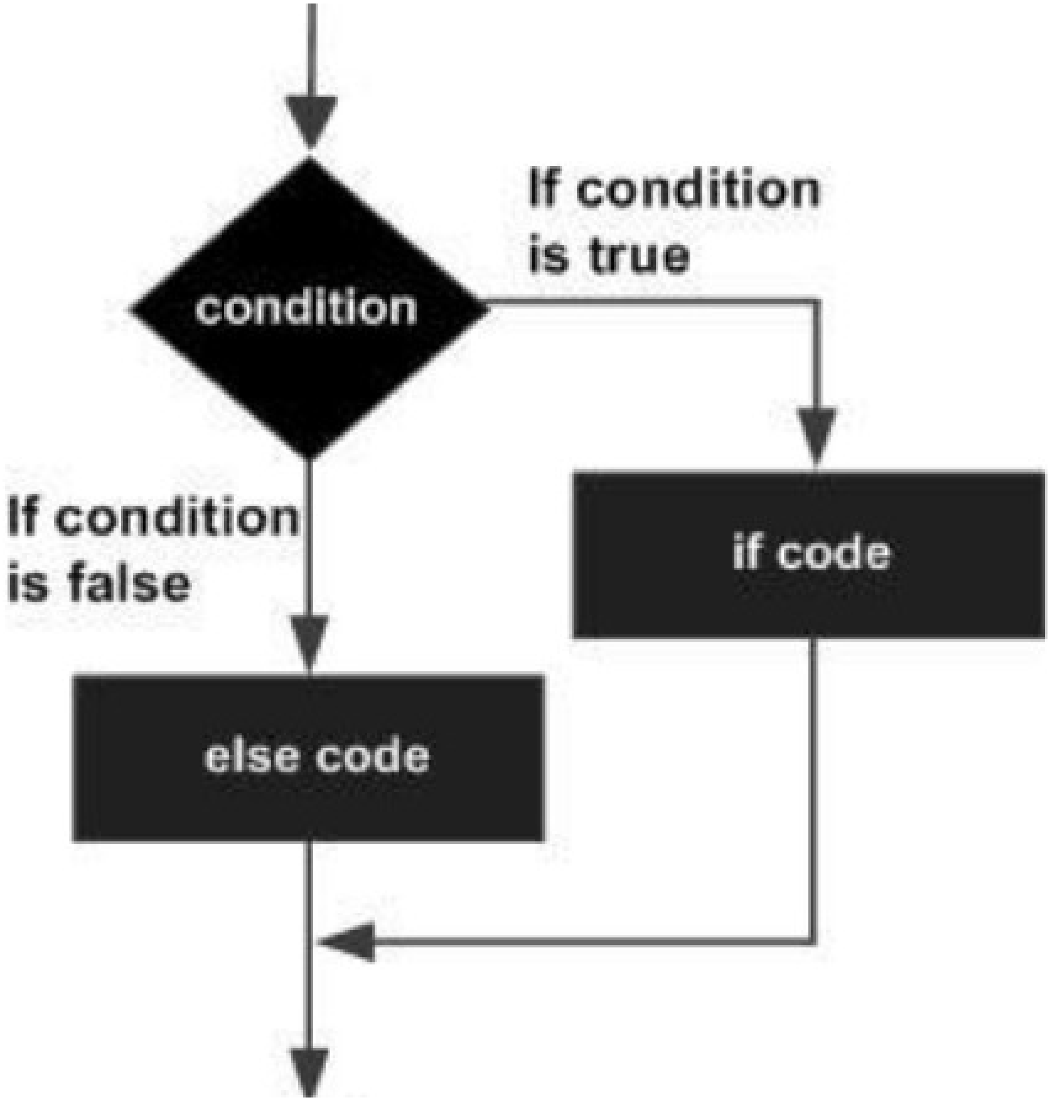
An if statement can be followed by an optional else statement, which executes when the boolean expression is false.

- Algorithm (pseudo-code) :

```
 if condition then
        processing1
else
        processing2
endif
```

if…else statement

Algorithm (flowchart):



if…else statement

“C” syntax;

```
if (codition)
{
        /* processing1 will be executed if condition is true */
 }
else
{
        /* processing2 will be executed if condition is false */
 }
```

if...else statement

Example : compute and print the absolute value of an integer a

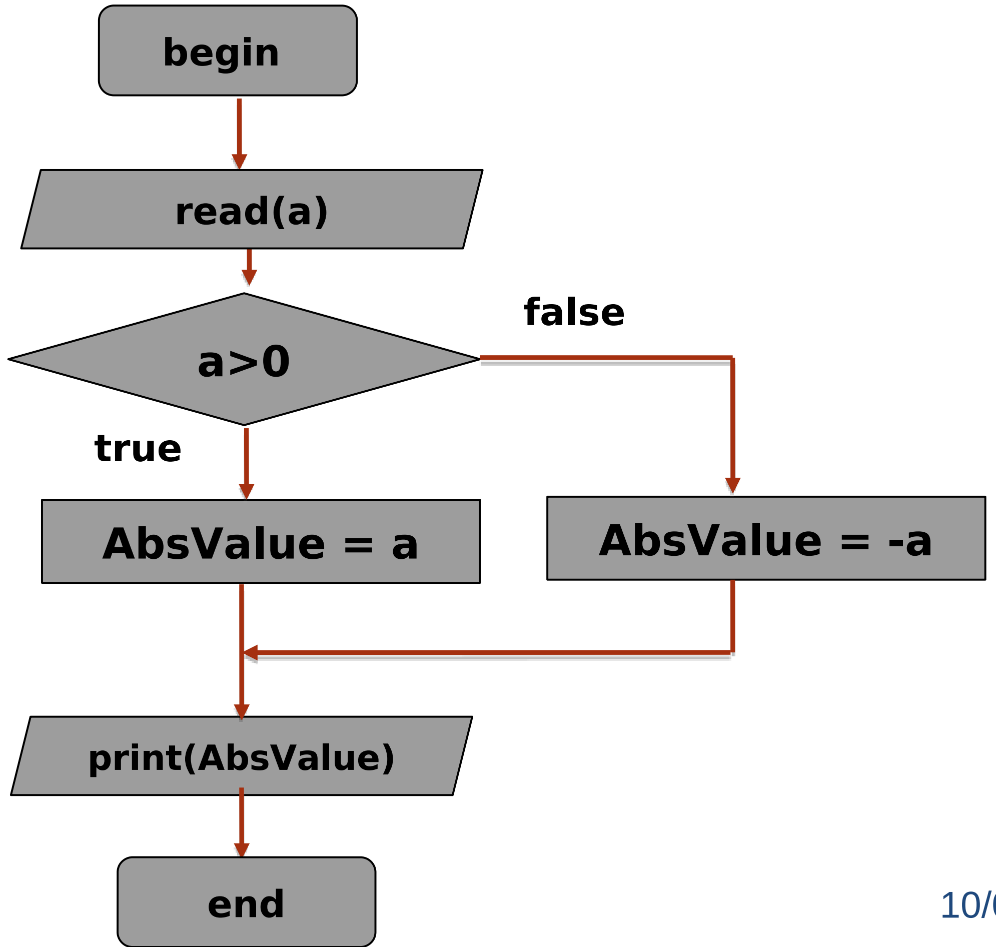
Algorithm:

```
begin
read (a)
if (a>0) then
        AbsValue = a
sinon
        AbsValue = -a
endif
print (val_abs)
end
```

if…else statement

Example : compute and print the absolute value of an integer a

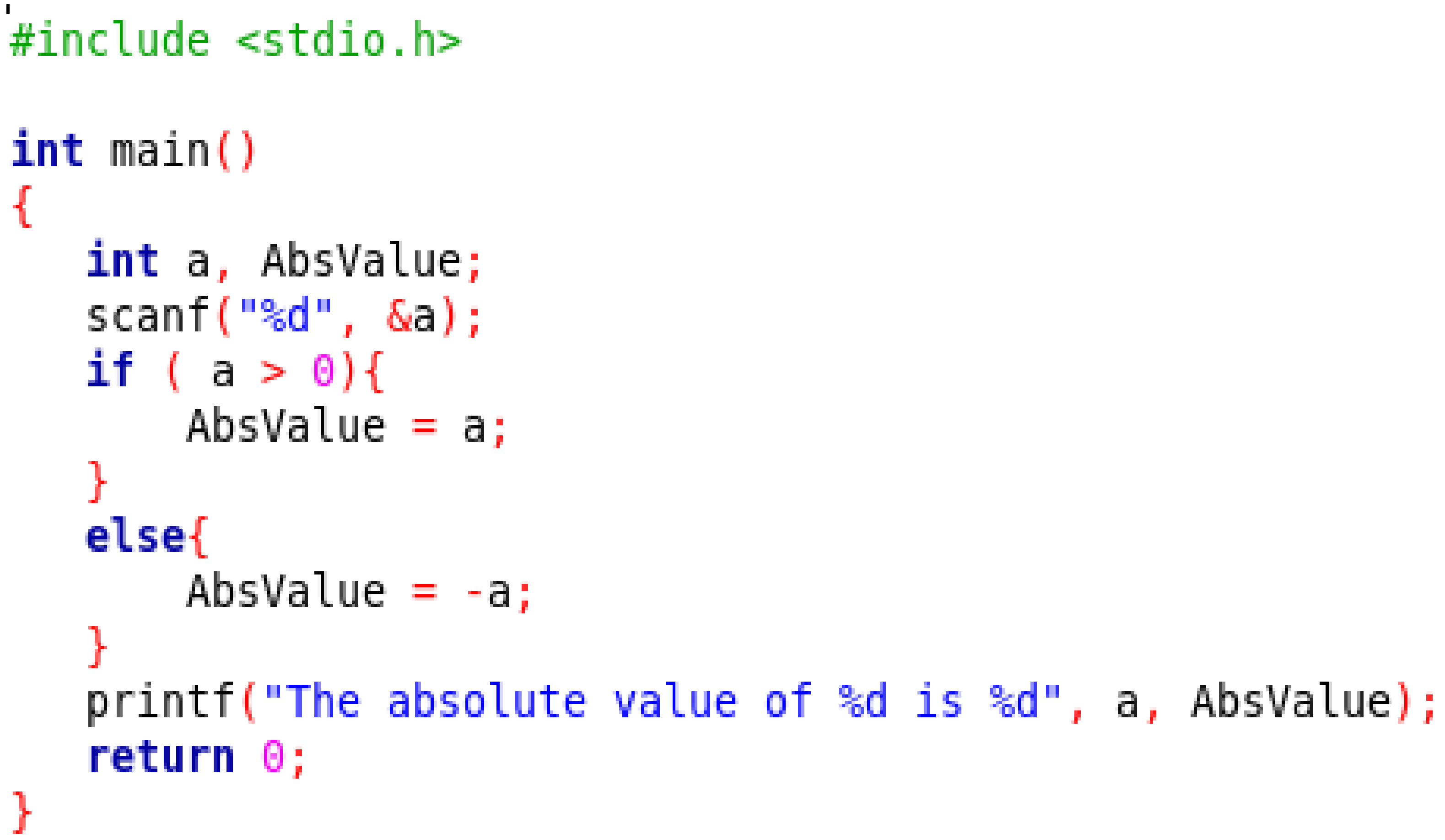
Flowchart:



if…else statement

Example : compute and print the absolute value of an integer a

C code:



```c
#include <stdio.h>

int main()
{
    int a, AbsValue;
    scanf("%d", &a);
    if ( a > 0){
        AbsValue = a;
    }
    else{
        AbsValue = -a;
    }
    printf("The absolute value of %d is %d", a, AbsValue);
    return 0;
}
```

The if...else if...else Statement

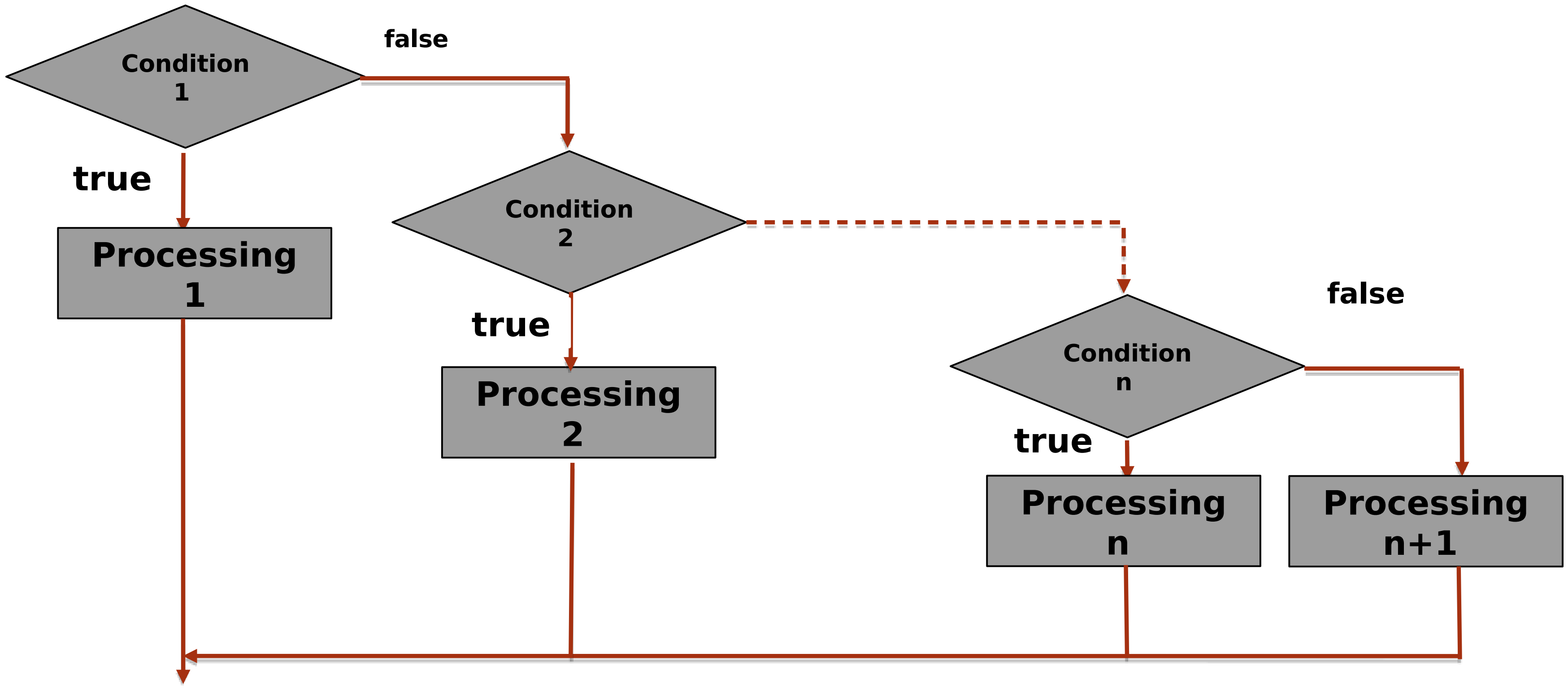
An if statement can be followed by an optional else if...else statement, which is very useful to test various conditions using single if...else if statement.

Pseudo-code:

```
if condition1 then
                    Processing 1
else if condition2 then
                   Processing 2
...
else if condition n alors
                 Processing n
else
                 Processing1 n+1
endif
```

The if...else if...else Statement

- **Flowchart:**



The if...else if...else Statement

“C” syntax:

```
if condition 1 {
          //Processing 1
}
else if condition 2 {
         // Processing 2
}
...
else if condition n {
         //Processing n
else{
         //Processing n+1
}
```

The if…else if…else Statement

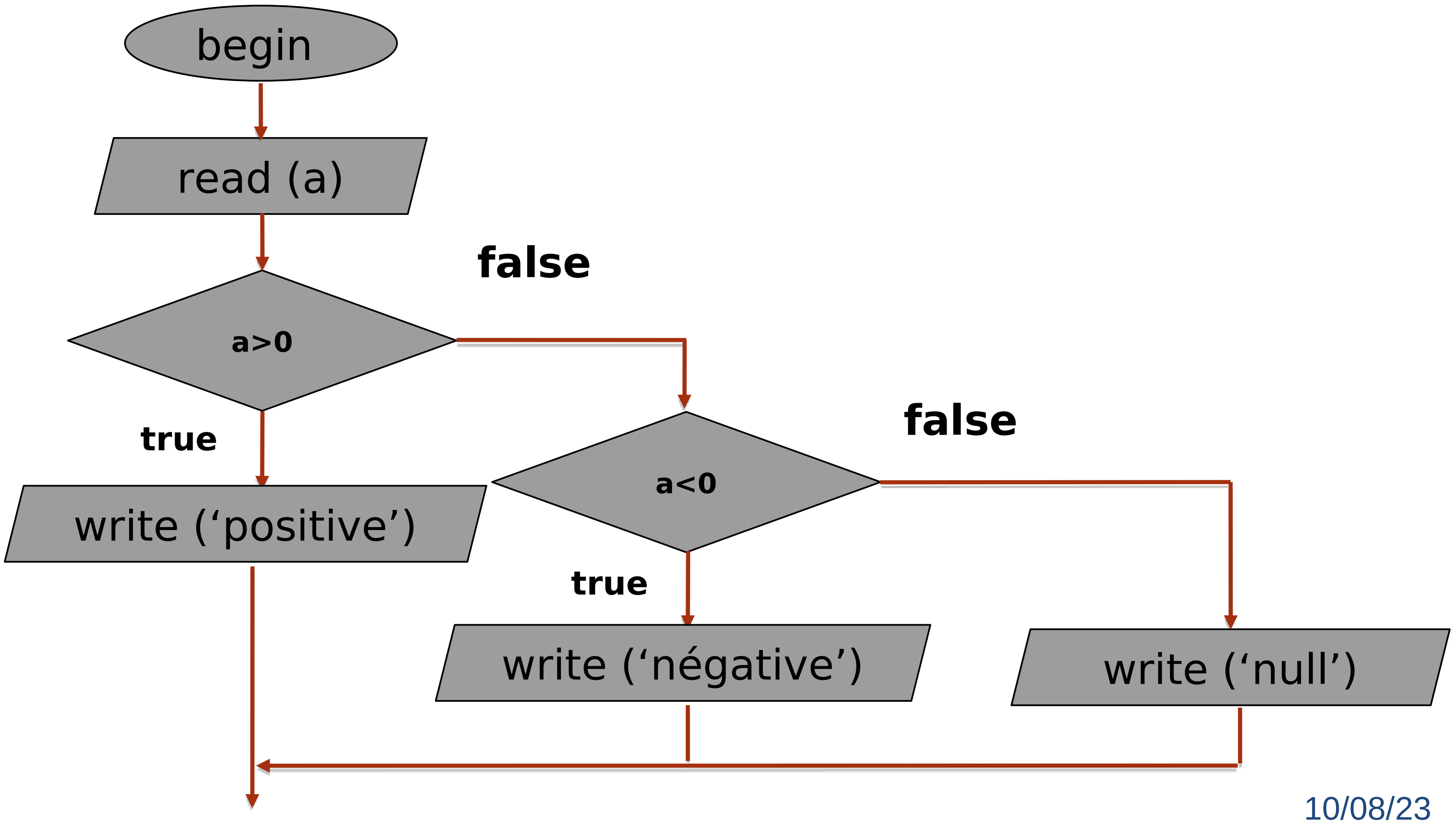
- **Example** : find the sign of a number ‘a’

- Algorithm

```
begin
read(a)
if a>0 then
                    write (‘positive’)
else if a<0 then
                    write (‘negative’)
else
          write (‘null’)
Endif
write(“it over”)
ense
```

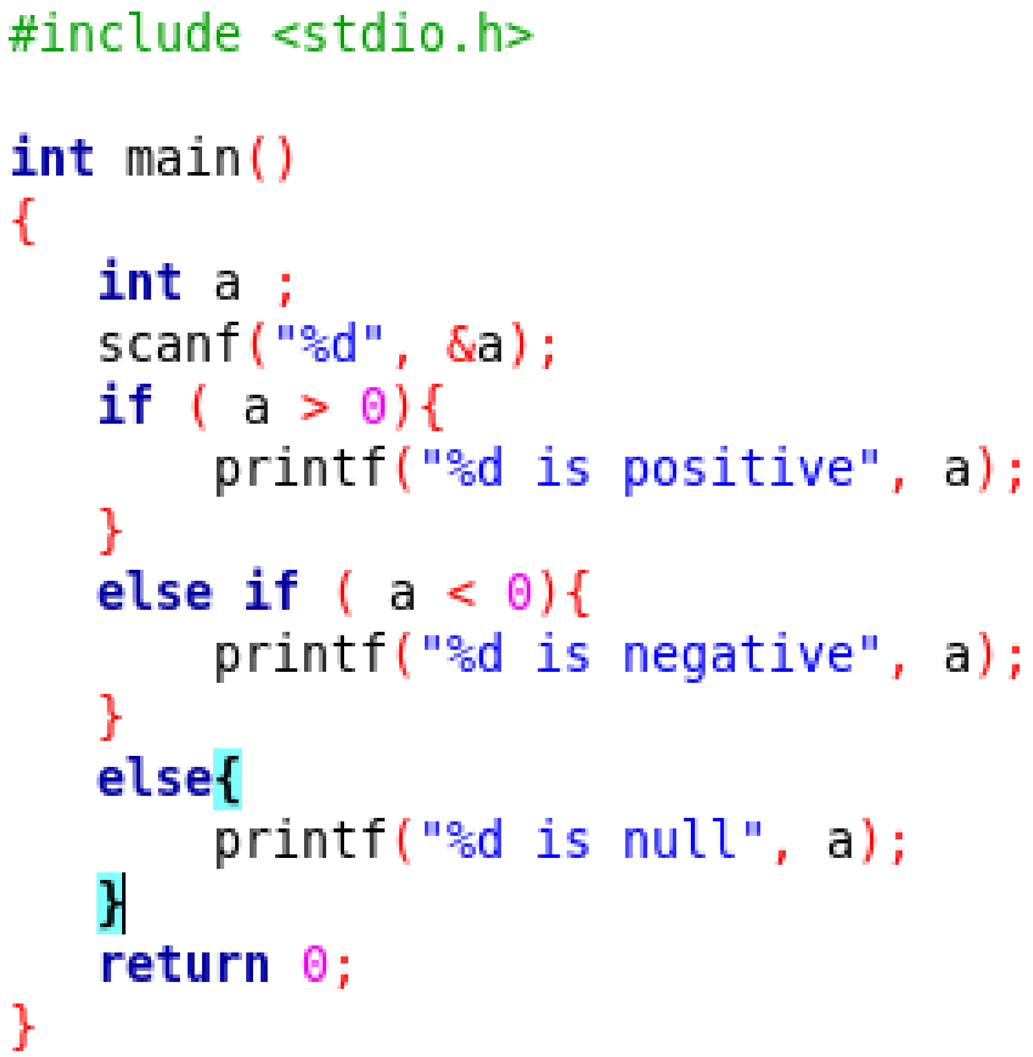
The if…else if…else Statement

Flowchart:



The if…else if…else Statement

C Code :

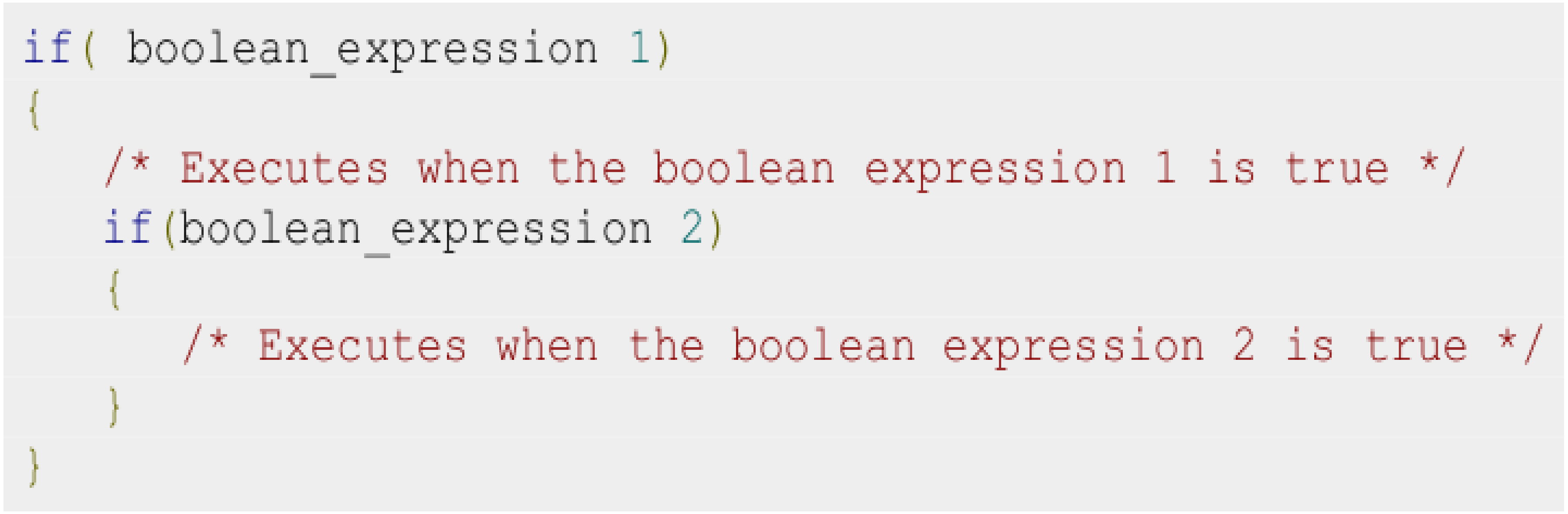


```c
#include <stdio.h>

int main()
{
    int a ;
    scanf("%d", &a);
    if ( a > 0){
        printf("%d is positive", a);
    }
    else if ( a < 0){
        printf("%d is negative", a);
    }
    else{
        printf("%d is null", a);
    }
    return 0;
}
```

Nested if statements

It is always legal in C programming to nest if-else statements, which means you can use one if or else if statement inside another if or else if statement(s).

Syntax:



```
if( boolean_expression 1)
{
   /* Executes when the boolean expression 1 is true */
   if(boolean_expression 2)
   {
      /* Executes when the boolean expression 2 is true */
   }
}
```

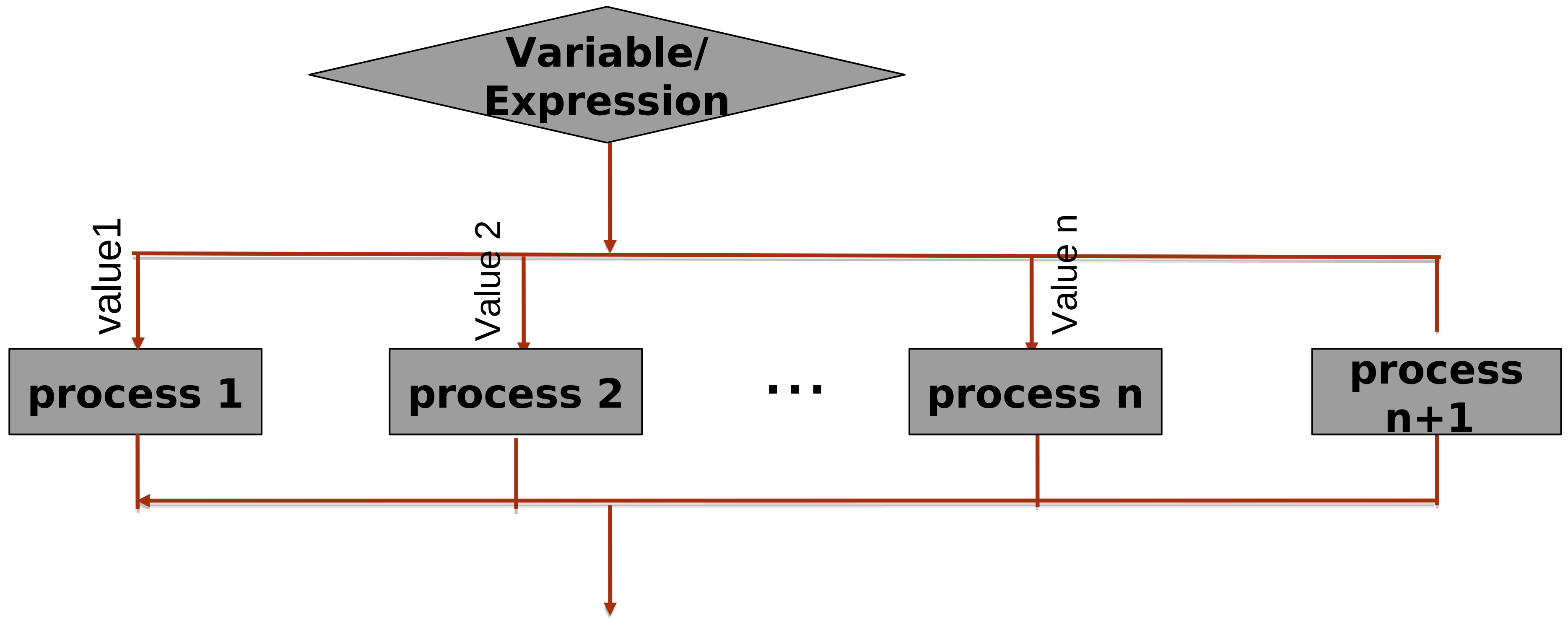
Nested if statements

Example: Find the greatest number among three numbers.

switch statement

A switch statement allows a variable or expression to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each switch case.

Flow chart



switch statement

“C” syntax:

```
switch(variable)
{
 case value1:
          //Processing 1 ;
          break;
 case valeur2:
          //Processing 2;
          break;
 …
 case valeur n:
          //Processing n;
          break;
 default:
          //Processing n+1;
}
```

switch statement

The following rules apply to a switch statement:

- The expression used in a switch statement must have an integer type.

- You can have any number of case statements within a switch. Each case is followed by the value to be compared to and a colon (:).

- The constant-expression for a case must be integer and it must be a constant or a literal.

- When the variable being switched on is equal to a case, the statements following that case will execute until a break statement is reached.

switch statement

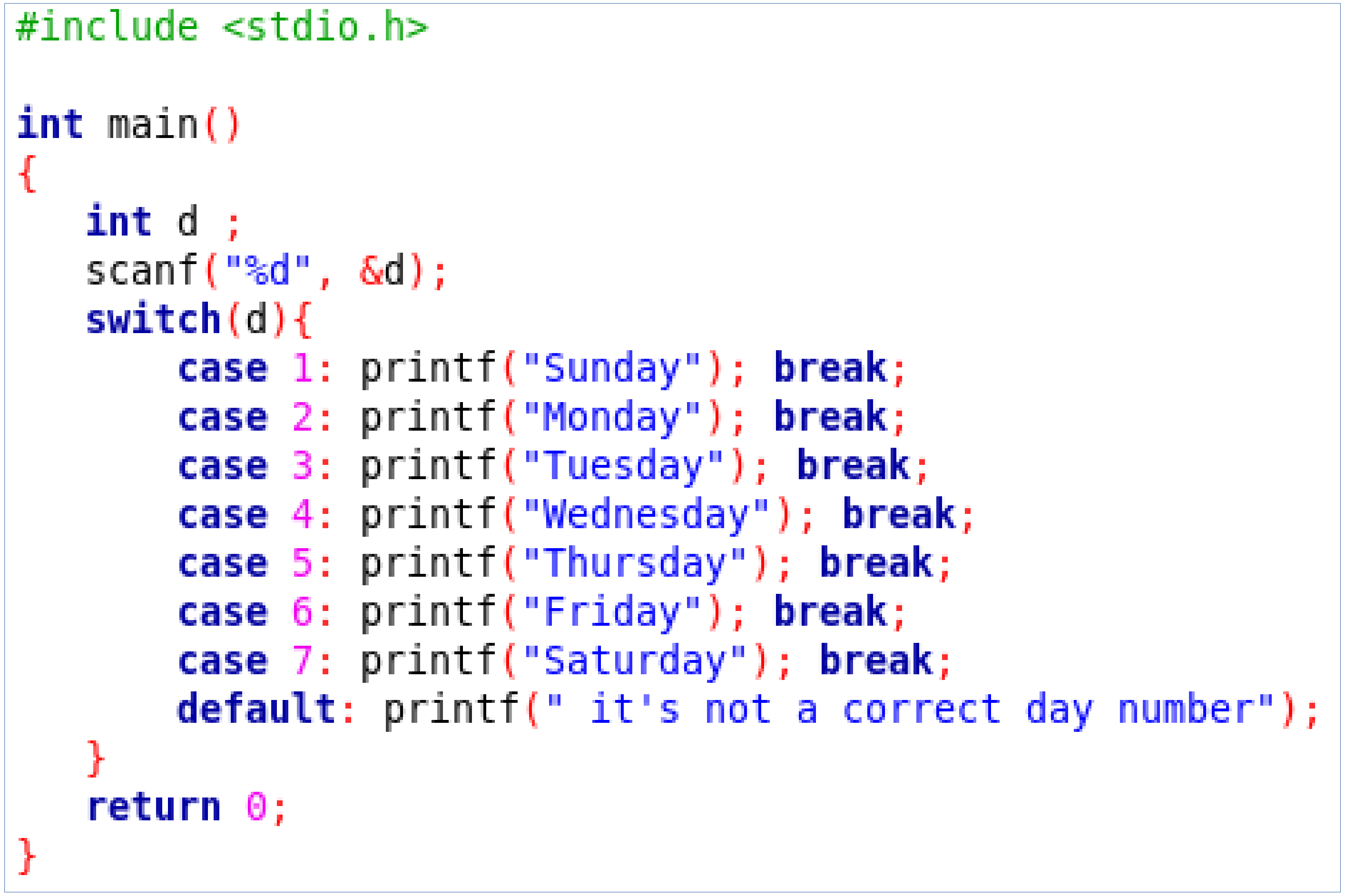
- When a break statement is reached, the switch terminates, and the flow of control jumps to the next line following the switch statement.

- Not every case needs to contain a break. If no break appears, the flow of control will fall through to subsequent cases until a break is reached.

- A switch statement can have an optional default case, which must appear at the end of the switch. The default case can be used for performing a task when none of the cases is true. No break is needed in the default case.

switch statement

Example: Write a C program to read a number representing days of the week (1 . 7) and using switch statement print the corresponding day.



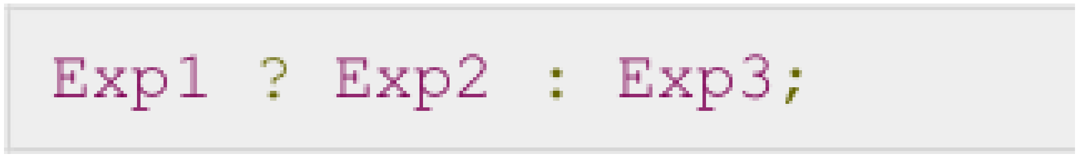
```c
#include <stdio.h>

int main()
{
   int d ;
   scanf("%d", &d);
   switch(d){
       case 1: printf("Sunday"); break;
       case 2: printf("Monday"); break;
       case 3: printf("Tuesday"); break;
       case 4: printf("Wednesday"); break;
       case 5: printf("Thursday"); break;
       case 6: printf("Friday"); break;
       case 7: printf("Saturday"); break;
       default: printf(" it's not a correct day number");
   }
   return 0;
}
```

Nested switch statements

It is possible to have a switch as part of the statement sequence of an outer switch. Even if the case constants of the inner and outer switch contain common values, no conflicts will arise.

The conditional operator (?:)

The conditional operator can be used to replace **if...else** statements. It has the following general form:



```
Exp1 ? Exp2 : Exp3;
```

Where Exp1, Exp2, and Exp3 are expressions. Notice the use and placement of the colon.

The value of a ? expression is determined like this: Exp1 (conditional expression) is evaluated. If it is true, then Exp2 is evaluated and becomes the value of the entire ? expression. If Exp1 is false, then Exp3 is evaluated and its value becomes the value of the expression.

The conditional operator (?:)

Example : Compute the absolute value of a number.

```
if (n > 0)
        abs_n = n ;
else
        abs_n = -n ;
```

This can be written as:

```
abs_n = ( n > 0) ? n :-n ;
```