2. S. Srivasthava]

Lesson Number : 17

Control Statements :

→ In C programs, statements are executed sequentially in the order in which they appear in the program.

→ But sometimes we may want to use a condition for executing only a part of program. Also many situation's arise where we may want to execute some statements several times.

→ Control statements enables us to specify the order in which the various instructions in the program are to be executed. This determines the flow of the control.

→ Control statements defines how the control is transfered to other parts of the program.

→ C language supports several types of control statements such as if statement, if...else statement, goto statement, switch statement etc. These statements are also called as decision-making statements.

Compound Statement or Block :

→ A Compound statement or a block is a group of statements enclosed within a pair of curly braces {}.

→ The statements inside the block are executed sequentially.

→ The general form is ;

```
{
  Statement 1 ;
  Statement 2 ;
  --------
  --------
}
```

→ For example ;

```
{
     l = 4 ;
     b = 2 ;
  area = l * b ;
  printf ( " %d", area) ;
}
```

→ A compound statement is syntactically equivalent to a single statement and can appear anywhere in the program where a single statement is allowed.

Decision Making with if statement :

→ The statements are also known as Decision-making statements. because they control the flow of execution. The statements are also known as control statements.

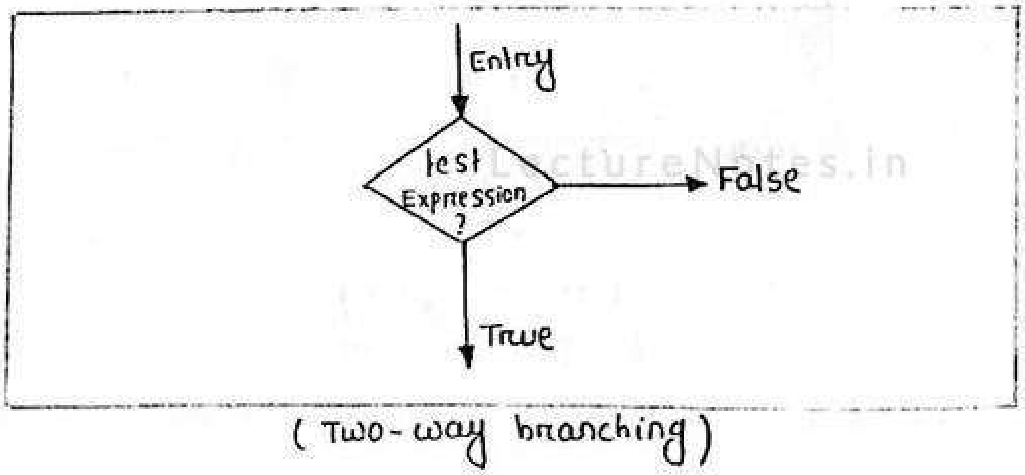
→ The if statement is a powerful decision-making statement and is used to control the flow of execution of statements.

→ It is basically a two-way decision statement and is used in conjunction with an expression.

→ It takes the form ; if (test expression)

→ It allows the computer to evaluate the expression first and then depending on whether the value of the expression (relation or condition) is 'true' (or non-zero) or 'false' (zero), it transfers the control to a particular statement.

→ This point of program has two paths to follow, one for the true condition and the other for the false condition, as in figure below.



(Two-way branching)

→ The if statement may be implemented in different forms depending on the complexity of condition to be tested.

→ The different forms are ;

1. Simple if statement .
2. if ... else statement .
3. Nested if ... else statement .
4. else if ladder .

1) Simple if Statement :

→ C uses the if statement to execute a set of command lines or one command line when the logical condition is true .

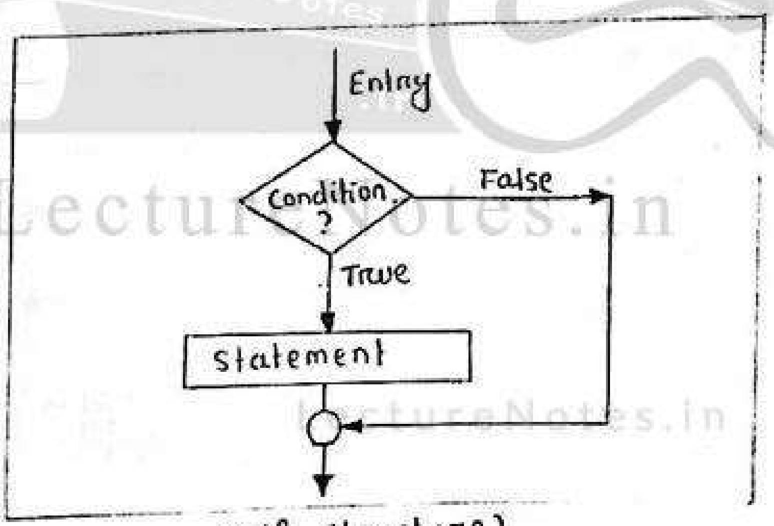
→ It has only one option .

→ The set of command lines or command lines are executed only when the logical condition is true .

→ Syntax for the simplest if statement ;

```
if ( condition)
      statement ;
```

→ Flowchart :



(if structure)

Example :

/* write a program to check whether the entered number is less than 10 ? if yes display the same */

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

```
void main( )
{
  int a ;
  Clrscr();
  printf (" Enter the number :");
  scanf ("%d", &a);
  if (a < 10)
     printf ("\n Number entered is less than 10");
     getch();
}
```

output :
```
Enter the number : 8
Number entered is less than 10
```

Example : /* write a program to check equivalence of two numbers */

```
#include <stdio.h>
#include <conio.h>
void main()
 {
  int a,b ;
  Clrscr();
  printf ("\n Enter Two Numbers :");
  Scanf ("%d %d", &a, &b);
  if (a-b == 0)
     printf ("\n Two Numbers are equal");
     getch();
 }
```

output :
```
Enter Two Numbers : 9 9
Two Numbers are equal .
```

The if.....else statement :

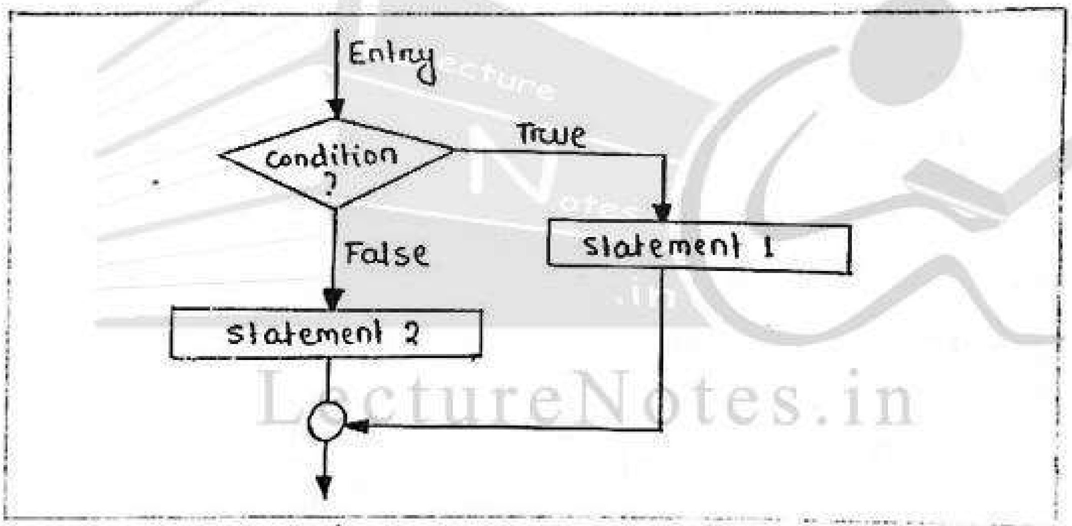
The if statement executes only when the condition following if is true. It does nothing when the codition is false.

The if...else statement takes care of true as well as false conditions. It has two blocks.

one block for if and it is executed when the condition is true. The other block is of else and it is executed when the condition is false.

The else statement cannot be used without if. No multiple else statements are allowed with one if.

Flowchart of if..else statement is ;



(if...else structure)

Syntax of if..else statement is ;

```
if ( codition is true)
    {
      statement 1 ;
    }
else
     { statement 2 ;
     }
```

Example : /* program to find the roots of a quadratic equation */

```
#include<stdio.h>
# include <conio.h>
# include <math.h>
void main()
 {
   int  a,b,c ;
   float x1, x2;
   clrscr();
   printf ("\n Enter values for a,b.c :");
   scanf (" %d %d %d", &a, &b, &c);
 if ( b*b > 4 *a *c)
  {
    x1 = -b + sqrt ( b*b-4 *a *c)/2*a ;
    x2 = -b - sqrt (b*b - 4 *a *c) / 2 *a ;
  printf (" \n x1 = %f  x2 = %f ", x1 , x2);
  }
 else
    {
     printf ( "\n Roots are Imaginary");
    }
 getch();
}
```

output :

```
Enter the values for a,b,C : 5 1 5

        Roots are Imaginary
```

Example : /* program to find the greatest between two numbers */

```
#include <stdio.h>
#include <conio.h>
void main()
 {
   int a,b;
   clrscr();
   printf ("\n Enter two numbers");
   scanf ("%d %d", &a, &b);
  if (a > b)
     { printf (" %d is greater than %d", a,b);
     }
  else
     { printf (" %d is greater than %d", b,a);
     }
  getch();
 }
```

<u>output</u> :

```
Enter two numbers : 5 10
10 is greater than 5 .
```

3) <u>Nested if--else statement</u> :

→ In this kind of statements number of logical conditions are checked for executing various statements.

→ Here, if any logical condition is true, the compiler executes the block followed by if condition otherwise it skips and executes else block.

→ In if...else statement else block is executed by default after failure of condition.

→ In order to execute the else block depending upon certain condition we can add repetitively if statements in else block.

→ This kind of nesting will be unlimited.

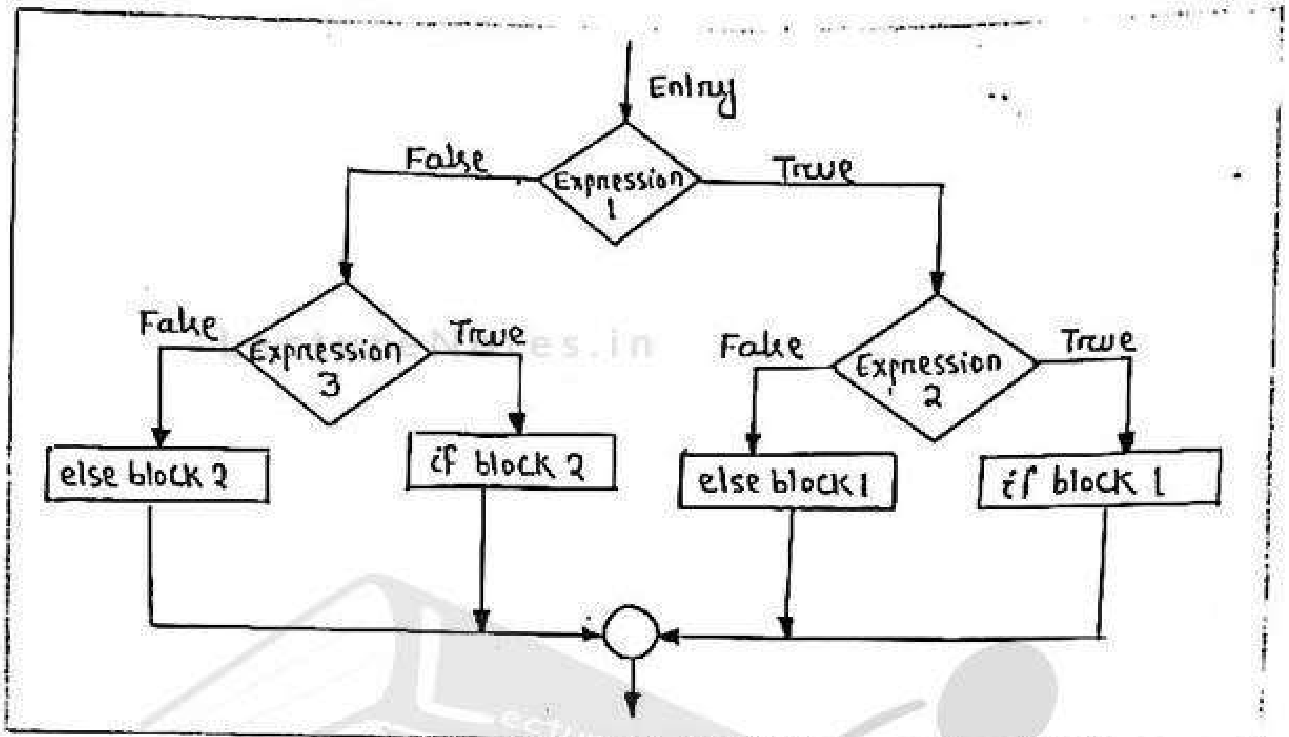
→ The syntax of if..else...if ladder is ;

```
if ( expression1)
  {
   if (expression 2)
    {
     ----------
     /* if block1 */
     ----------
    }
   else
     {
      --------
      /* else block1 */
      --------
     }
  }
else
  {
   if (expression 3)
    {
     ------
     /* if block 2 */
     -------
    }
   else
     {
      --------
      /* else block 2 */
     }
  }
```

→ If the expression 1 is true, the expression 2 will be evaluated, if it is true, the if block 1 will be executed, otherwise else block1 will be executed.

→ If the expression1 is false. the expression 3 will be evaluated. if it is true, the if block 2 will be executed. otherwise else block2 will be executed ..

→ Flowchart :



(Nested if...else statement)

Example : /* program to find the largest between three numbers*/

```
#include <stdio.h>
#include < conio.h>
void main( )
 {
   int  x, y, z ;
   clrscr( );
   printf (" Enter three numbers :");
   scanf ( "%d%d%d", &x , &y , &z);

  if ( x>y )
    {
     if ( x>z )
        printf(" \n Largest  = %d", x) ;
    else
        printf ( " \n Largest = %d", z ) ;
    }
```

Assignment : what is a statement ? what is the control statement?

→ Statement : The executable part of the C programs which are executed sequentially in the order in which they appear in the program is called statement .

→ Control statement : control statements enables us to specify the order in which various instructions in the program are to be executed . This determines the flow of control. The control statement also defines how control is transferred to other parts of the program.

Assignment : Define Compound statement .

→ Compound statement : A compound statement or block is a group of statements enclosed within a pair of curly braces {}. The statements inside the block are executed sequentially.

→ The general form of compound statement is ;

```
{
   statement 1 ;
   statement 2 ;
   - - - - - -
   - - - - -
}
```

→ Example :

```
{
   l = 4 ;
   b = 5 ;
area = l * b ;
printf ("%d", area);
}
```

Assignment : write a program to check whether the candidate's age is greater than 17 or not. If yes, display message " Eligible".

```
#include <stdio.h>
#include <conio.h>
void main()
  {
    int age;
    clrscr();
    printf(" \n Enter age :");
    scanf ("%d", &age);
   if ( age > 17)
     {
       printf (" Eligible");
     }
  getch();
}
```

output :
```
Enter age : 19
Eligible
```

Assignment : write a pgm to enter only three no. & calculate their sum.

```
# include <stdio.h>
# include <conio.h>
void main( )
  { int a,b,c,x;
    clrscr();
    printf( "Enter 3 no.\n");
 x = scanf ("%d %d %d", &a ,&b,&c);
if ( x==3)
     { printf ("\n Addition : %d", a+b+c);
     }
 getch( );
}
```

```
else
  if ( Y > z)
     printf ( "\n Largest = %d", Y);
 else
    printf ( "\n Largest = %d", z );
 }
getch();
}
```

output : Enter three numbers : 30 20 60
Largest = 60

[References : 1. S.Srivasthava
2. E.Balagurusami]

Lesson Number : 18

4) The else if ladder :

→ Sometimes we wish to make a multi-way decision based on several conditions. The most general way of doing this is by using the else if variant on the if statement. This works by cascading several comparisons.

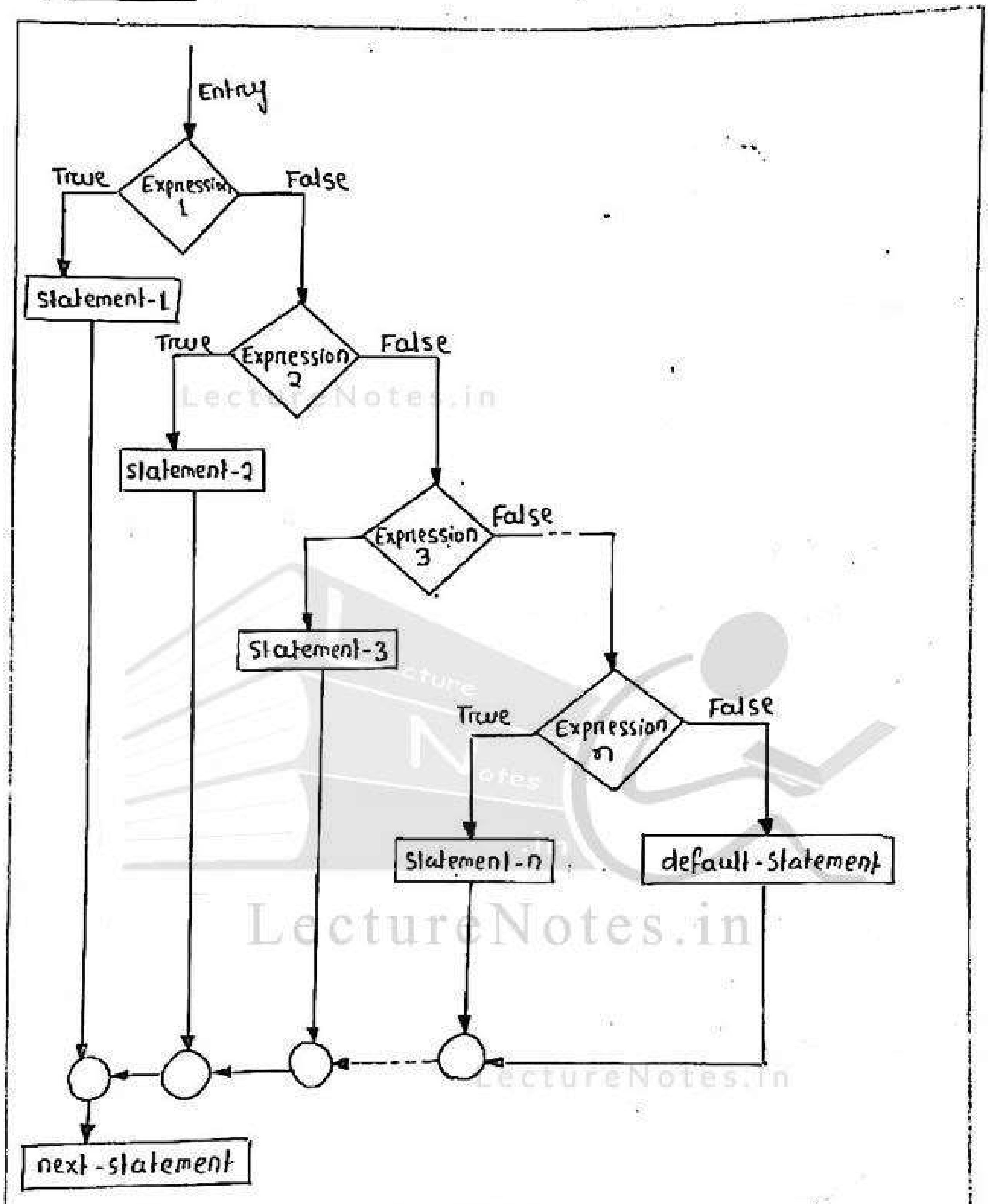
→ As soon as one of these gives a true result, the flowing statement or block is executed and no further comparisons are performed.

→ It consists of chain of ifs in which the statement associated with each else is an if.

→ The Syntax for else if ladder is ;

```
if (Expression 1 )
      statement 1 ;
else if ( Expression 2)
         statement 2 ;
   else if (Expression 3)
           Statement 3 ;
          ----------
          ----------
else if (expression n)
         statement n;
else   default - statement;
   next statement ;
```

→ Flowchart :



(else if ladder)

→ Here the expressions are evaluated from the top of the ladder, downwords. If the execution expression results non-zero, the statement associated are executed and the control is transferred to next statement skipping the rest of the ladder.

→ when all expressions (n) results false , then the final else i.e. the default statement is executed .

Example : /* program for awarding grades depending on the examination result */

```
# include <stdio.h>
# include <Conio.h>
 void main( )
 {
    int result ;
    Clrscr( );
    printf ( " Enter the mark obtained :\n") ;
     Scanf ( "%d", & result ) ;
  if ( result >=90)
        printf (" passed : Grade O\n");
  else if (result >= 80)
        printf (" passed : Grade E\n");
  else if ( result >=70)
        printf (" passed : Grade A \n");
  else if ( result >= 60)
           printf ("passed : Grade B\n");
  else if ( result >=50)
            printf (" passed : Grade C\n");
  else if (result >= 40)
             printf (" passed : Grade D\n");
 else
        printf ( " Failed \n");
 }       getch();
```

Output :

```
Enter the mark oblained : 78
Passed : Grade A
```

Program :

```
/* Write a program to check whether a year is a leap year
   or not */

#include <stdio.h>
# include <conio.h>
void main ( )
{
     int year ;
    clrscr( );
  printf ( " \n Enter the year :");
  scanf ( "%d", &year);
if ( year % 100 != 0)
     {   if ( year % 400 = = 0)
             printf ( " The year is a leap year") ;
       else
             printf ( " The year is not a leap year");
     }
else
       {   if ( year % 400= = 0)
             printf ( " The year is a leap year") ;
          else
              printf ( " The year is not a leap year");
     }
getch( ) ;
}
```

✓ OR

```
#include <stdio.h>
#include <conio.h>
void main()
{
    int year;
    clrscr();
    printf(" Enter the year :\n");
    scanf( "%d", &year);
    if ((year % 400 == 0) || ((year % 100 != 0) && (year % 4 == 0)))
      {
        printf(" The year is a leap year");
      }
    else
      {
        printf(" The year is not a leap year");
      }
getch();
}
```

centurion year

1900 is not a leap year bt ~~is~~ ~~is~~ it is divisible by 4 & year % 100!=0 also satisfy that's why we have to check (year % 400 == 0) for leap year.

Output :

```
Enter the year : 2009
The year is not a leap year.
```

(*) → A year is called leap year if it is divisible by 400.

Exp: 1600, 2000 etc.

→ if yr is not divisible by 400 as well as 100 but it is divisible by 4 then that yr are also leap yr.

Exp: 2004, 2008, 2012