Decision Control Structure

Rab Nawaz Jadoon

Assistant Professor

COMSATS IIT, Abbottabad

Pakistan

Department of Computer Science

DCS

COMSATS Institute of
Information Technology

Decision control structure

- Before this we have used sequence control structure in which,
  - Various steps are executed sequentially, i.e. in the same order in which they appear in the program.
  - By default the instructions in a program are executed sequentially.

Decision control structure

- Many a times, we want a set of instructions to be executed in one situation, and an entirely different set of instructions to be executed in another situation.
  - This kind of situation is dealt in C programs using a decision control instruction.

Decision control structure

- Decision control instruction can be implemented in C using:
  - The if statement
  - The if-else statement
  - The conditional operators

The *If* Statement

- Like most languages, C uses the keyword if to implement the decision control instruction.
  - The general form of if statement looks like this:

    if (this condition is true)

        execute this statement ;

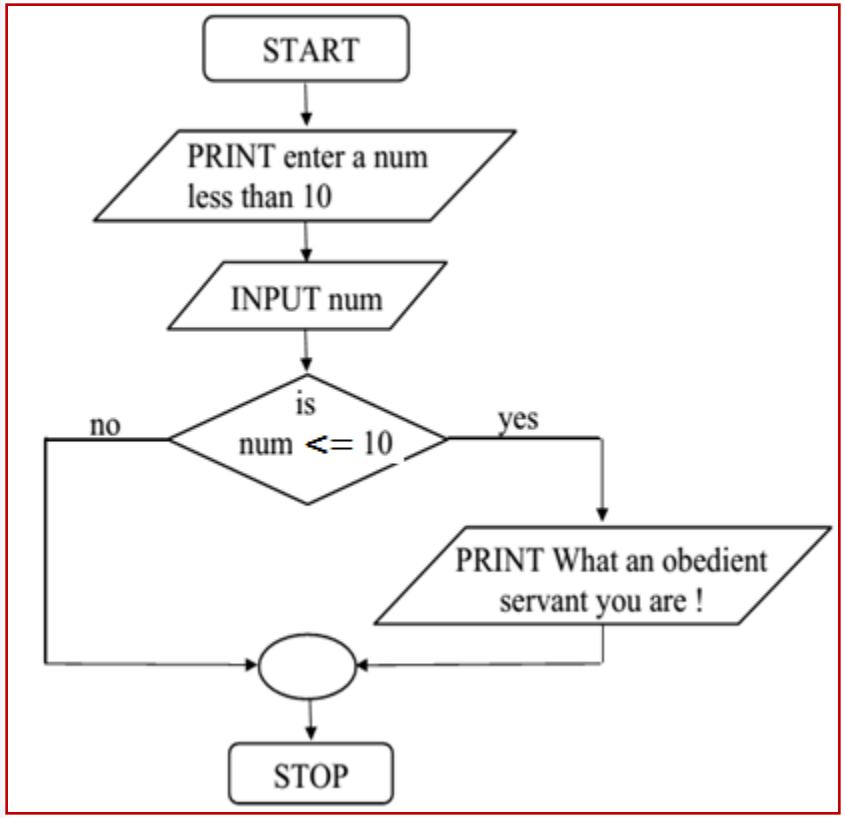
    - The condition following the keyword if is always enclosed within a pair of parentheses.
    - If the condition, whatever it is, is true, then the statement is executed.
    - If the condition is not true then the statement is not executed; instead the program skips past it.

The *If* Statement

- We express a condition using C's 'relational' operators.
  - The relational operators allow us to compare two values to see whether they are equal to each other, unequal, or whether one is greater than the other.

this expression	is true if
x == y	x is equal to y
x != y	x is not equal to y
x < y	x is less than y
x > y	x is greater than y
x <= y	x is less than or equal to y
x >= y	x is greater than or equal to y

Simple Program using *If* statement



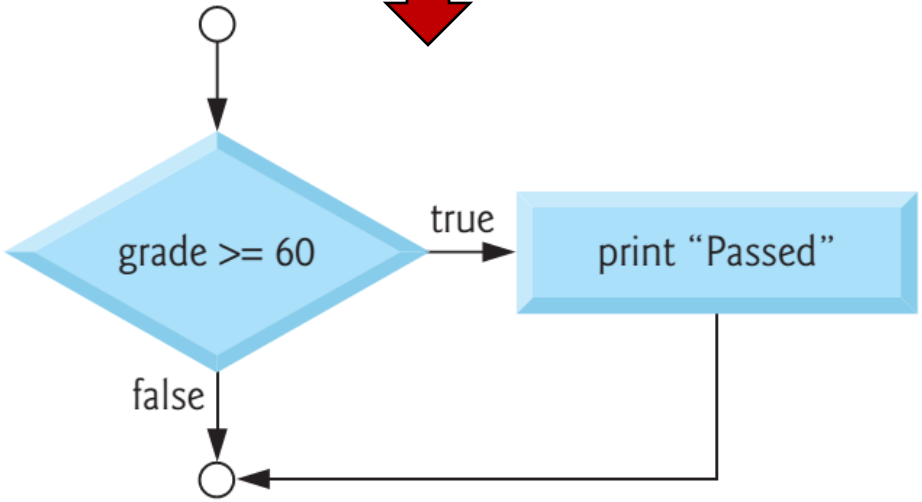

```
/* Demonstration of if statement */
main( )
{
      int   num ;

      printf ( "Enter a number less than 10 " ) ;
      scanf ( "%d", &num ) ;

      if ( num <= 10 )
            printf ( "What an obedient servant you are !" ) ;
}
```

If Control Structure

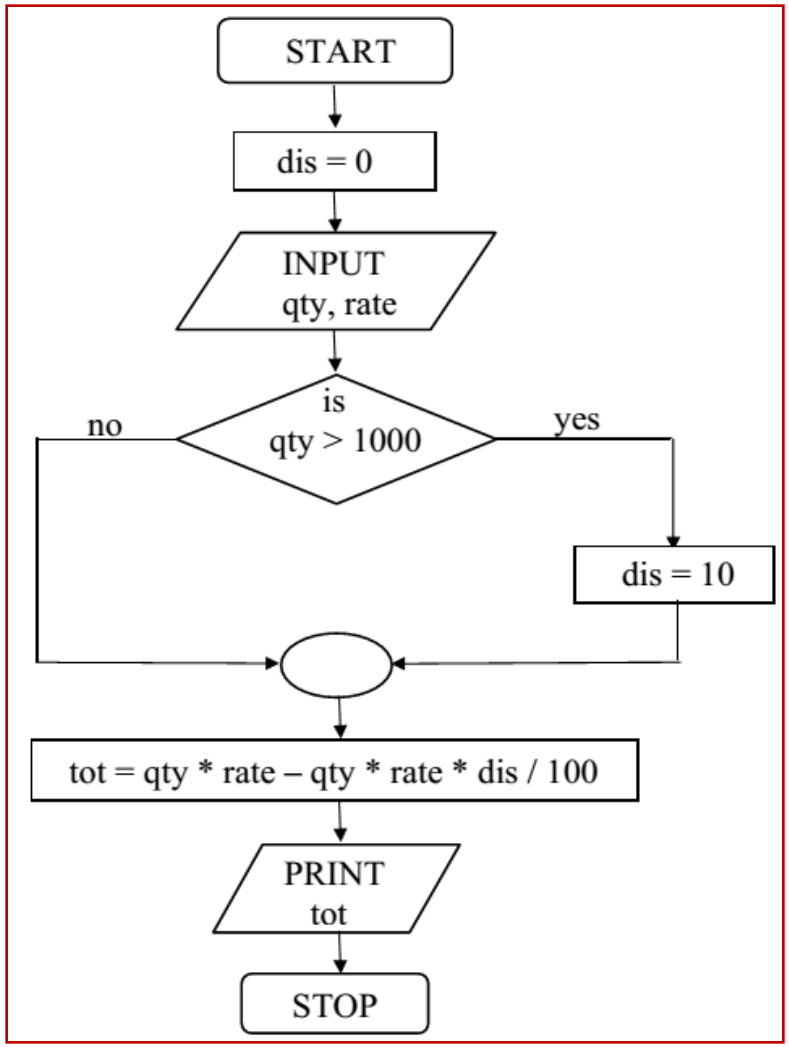
```
if ( grade >= 60 ) {
   printf( "Passed\n" );
} /* end if */
```



Problem

- While purchasing certain items, a discount of 10% is offered if the quantity purchased is more than 1000. If quantity and price per item are input through the keyboard, write a program to calculate the total expenses???

Solution (Flow Chart & Program)




```
/* Calculation of total expenses */
main( )
{
      int   qty, dis = 0 ;
      float   rate, tot ;
      printf ( "Enter quantity and rate " ) ;
      scanf ( "%d %f", &qty, &rate) ;

      if ( qty > 1000 )
            dis = 10 ;

      tot = ( qty * rate ) - ( qty * rate * dis / 100 ) ;
      printf ( "Total expenses = Rs. %f", tot ) ;
}
```

```
Enter quantity and rate 1200 15.50
Total expenses = Rs. 16740.000000

Enter quantity and rate 200 15.50
Total expenses = Rs. 3100.000000
```

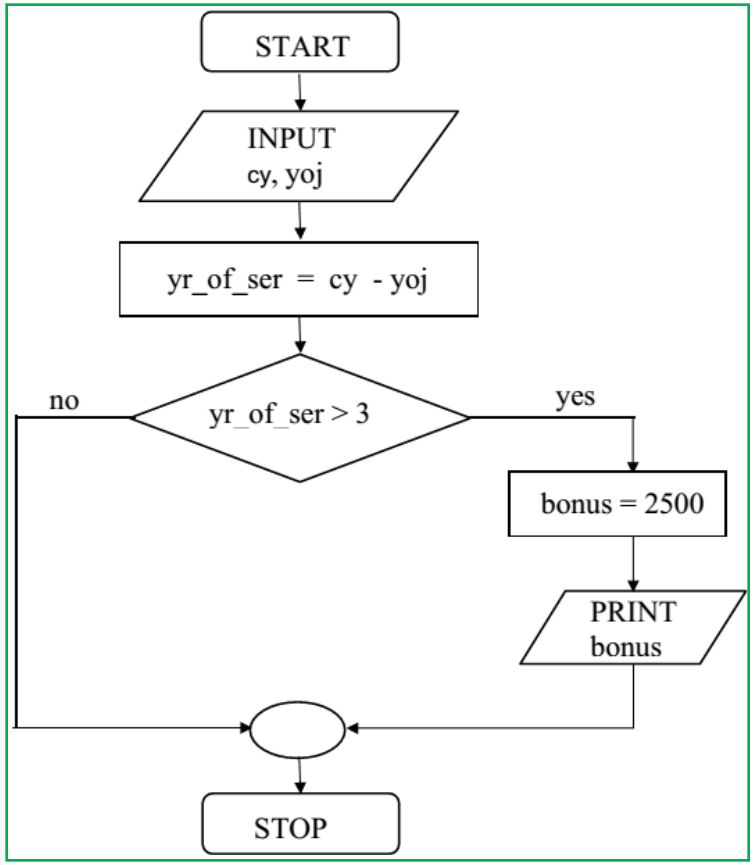
Multiple statements within *If*

- More than one statement to be executed if the expression following *if* is satisfied.
  - If such multiple statements are to be executed then they must be placed within a pair of braces.
  - For example,

```
If(condition)
{
---------
---------
}
```

Program

Flow chart



Program

```
/* Calculation of bonus */
main( )
{
      int   bonus, cy, yoj, yr_of_ser ;

      printf ( "Enter current year and year of joining " ) ;
      scanf ( "%d %d", &cy, &yoj ) ;

      yr_of_ser = cy - yoj ;

      if ( yr_of_ser > 3 )
      {
            bonus = 2500 ;
            printf ( "Bonus = Rs. %d", bonus ) ;
      }
}
```

Quiz

```
#include<stdio.h>
#include<conio.h>
 main( )
{
  int a = 300, b, c ;
     if ( a >= 400 )
     {
         b = 300 ;
         c = 200 ;
     printf ( "\n %d %d", b, c ) ;
     }
 getche ();
}
```

What would be the output???

NOTHING

What would be the output, if a=500???

300 200

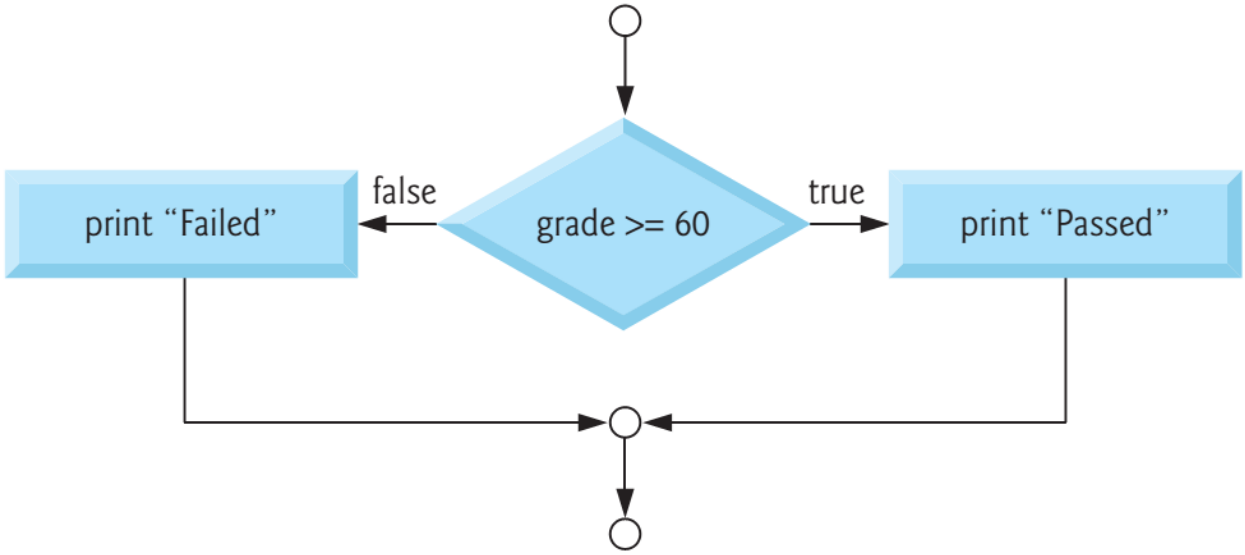
The *if-else* Structure

- Execute one group of statements if the expression evaluates to true and another group of statements if the expression evaluates to false.
  - This is done using the *if-else* control structure.
  - For example,

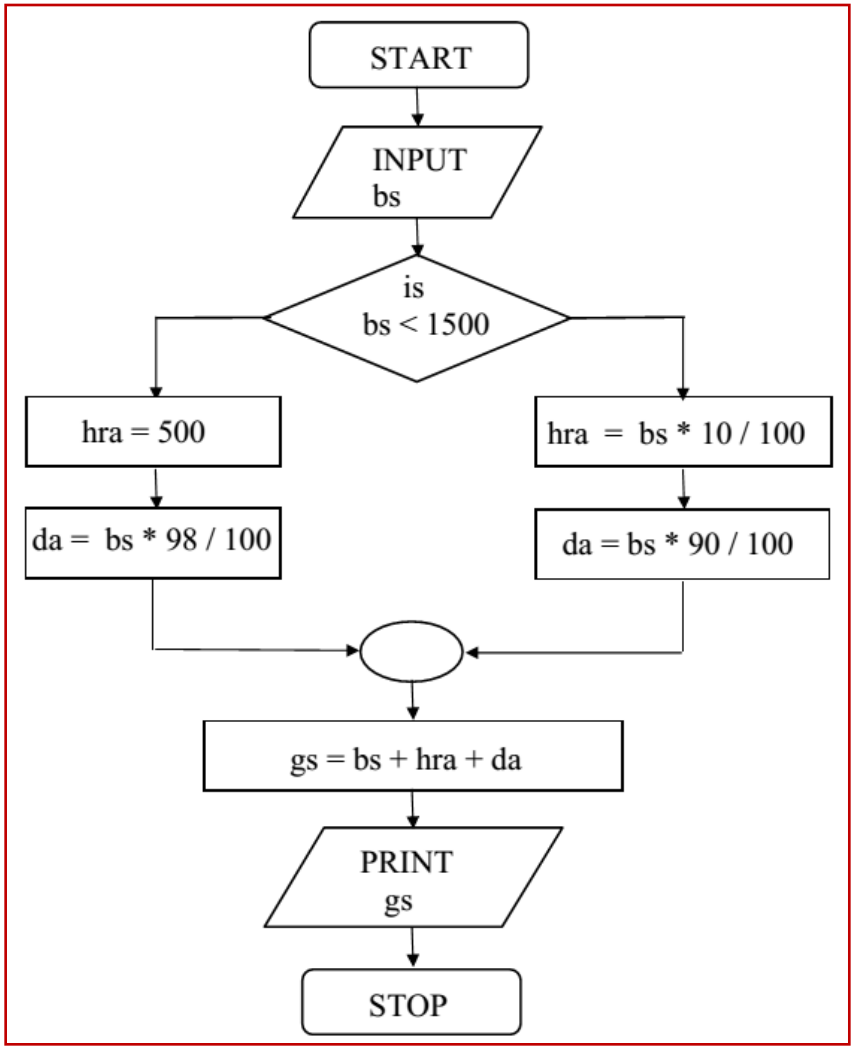
If student's grade is greater than or equal to 60
Print "Passed"
else
Print "Failed"

If-else Example

```
if ( grade >= 60 ) {
   printf( "Passed\n" );
} /* end if */
else {
   printf( "Failed\n" );
} /* end else */
```



Program illustrating *if-else*




```
/* Calculation of gross salary */
main( )
{
      float   bs, gs, da, hra ;

      printf ( "Enter basic salary " ) ;
      scanf ( "%f", &bs ) ;

      if ( bs < 1500 )
      {
            hra = bs * 10 / 100 ;
            da = bs * 90 / 100 ;
      }
      else
      {
            hra = 500 ;
            da = bs * 98 / 100 ;
      }

      gs = bs + hra + da ;
      printf ( "gross salary = Rs. %f", gs ) ;
}
```

Quiz ???

- What would be the output of the following program???

```
#include<stdio.h>
#include<conio.h>
main( )
{
 int x = 3 ;
 float y = 3.0 ;
 if ( x == y )
  printf ( "\nx and y are equal" ) ;
 else
  printf ( "\nx and y are not equal" ) ;

  getche();
}
```

x and y are equal

Quiz ???

- What would be the output of the following program???

```
#include<stdio.h>
#include<conio.h>
main( )
{
 int i = 65 ;
 char j = 'A' ;
 if ( i == j )
  printf ( "C is WOW" ) ;
 else
  printf( "C is a headache" ) ;
  getche();
}
```

C is WOW

Quiz ???

- Identify the error from the following

(a)
```
main( )
{
    float   a = 12.25, b = 12.52 ;
    if ( a = b )
        printf ( "\na and b are equal" ) ;
}
```

(b)
```
main( )
{
    int   j = 10, k = 12 ;
    if ( k >= j )
    {
        {
            k = j ;
            j = k ;
        }
    }
}
```

(c)
```
main( )
{
    if ( 'X' < 'x' )
        printf ( "\nascii value of X is smaller than that of x" ) ;
}
```

Identify the errors

(d)
```
main( )
{
    int  x = 10 ;
    if ( x >= 2 ) then
        printf ( "\n%d", x ) ;
}
```

(e)
```
main( )
{
    int  x = 10 ;
    if x >= 2
        printf ( "\n%d", x ) ;
}
```

(f)
```
main( )
{
    int  x = 10, y = 15 ;
    if ( x % 2 = y % 3 )
        printf ( "\nCarpathians" ) ;
}
```

Forms of *if*

(a)
```
if ( condition )
    do this ;
```

(b)
```
if ( condition )
{
    do this ;
    and this ;
}
```

(c)
```
if ( condition )
    do this ;
else
    do this ;
```

(d)
```
if ( condition )
{
    do this ;
    and this ;
}
else
{
    do this ;
    and this ;
}
```

(e)
```
if ( condition )
    do this ;
else
{
    if ( condition )
        do this ;
    else
    {
        do this ;
        and this ;
    }
}
```

(f)
```
if ( condition )
{
    if ( condition )
        do this ;
    else
    {
        do this ;
        and this ;
    }
}
else
    do this ;
```

Use of logical operators

- C allows usage of three logical operators, namely, &&, || and !.
  - These are to be read as 'AND', 'OR' and 'NOT' respectively.
  - two of them are composed of double symbols:
    - || and &&.
    - Don't use the single symbol | and &. These single symbols also have a meaning. They are bitwise operators.
    - The first two operators, && and ||, allow two or more conditions to be combined in an if statement.

Working of three logical operators

Operands		Results			
x	y	!x	!y	x && y	x || y
0	0	1	1	0	0
0	non-zero	1	0	0	1
non-zero	0	0	1	0	1
non-zero	non-zero	0	0	1	1

Example

- The marks obtained by a student in 5 different subjects are input through the keyboard.
  - The student gets a division as per the following rules:
    - Percentage above or equal to 60 - First division
    - Percentage between 50 and 59 - Second division
    - Percentage between 40 and 49 - Third division
    - Percentage less than 40 - Fail
  - Write a program to calculate the division obtained by the student.

Solution (Method-1)

```
main( )
{
      int   m1, m2, m3, m4, m5, per ;

      printf ( "Enter marks in five subjects " ) ;
      scanf ( "%d %d %d %d %d", &m1, &m2, &m3, &m4, &m5 ) ;
      per = ( m1 + m2 + m3 + m4 + m5 ) / 5 ;
      if ( per >= 60 )
            printf ( "First division ") ;
      else
      {
            if ( per >= 50 )
                  printf ( "Second division" ) ;
            else
            {
                  if ( per >= 40 )
                        printf ( "Third division" ) ;
                  else
                        printf ( "Fail" ) ;
            }
      }
}
```

Problems faced in the program

- This is a straight forward program. Observe that the program uses nested if-elses.
  - This leads to three disadvantages:
    - As the number of conditions go on increasing the level of indentation also goes on increasing. As a result the whole program creeps to the right.
    - Care needs to be exercised to match the corresponding ifs and elses.
    - Care needs to be exercised to match the corresponding pair of braces.

    - **Note:** All these three problems can be eliminated by usage of 'Logical operators'. (Method 2)

Solution (Method-2)

```
main( )
{
      int  m1, m2, m3, m4, m5, per ;

      printf ( "Enter marks in five subjects " ) ;
      scanf ( "%d %d %d %d %d", &m1, &m2, &m3, &m4, &m5 ) ;
      per = ( m1 + m2 + m3 + m4 + m5 ) / 5 ;

       if ( per >= 60 )
            printf ( "First division" ) ;

       if ( ( per >= 50 ) && ( per < 60 ) )
            printf ( "Second division" ) ;

       if ( ( per >= 40 ) && ( per < 50 ) )
            printf ( "Third division" ) ;

       if ( per < 40 )
            printf ( "Fail" ) ;
}
```

The *Else if Clause*

- There is one more way in which we can write the previous program,

```
main( )
{
      int  m1, m2, m3, m4, m5, per ;

      printf ( "Enter marks in five subjects " ) ;
      scanf ( "%d %d %d %d %d", &m1, &m2, &m3, &m4, &m5 ) ;
      per = ( m1 + m2 + m3 + m4 + m5 ) / 5 ;
      if ( per >= 60 )
            printf ( "First division" ) ;
      else if ( per >= 50 )
            printf ( "Second division" ) ;
      else if ( per >= 40 )
            printf ( "Third division" ) ;
      else
            printf ( "fail" ) ;
}
```

The *Else if Clause*

- Note that the ***else if*** clause is nothing different.
  - It is just a way of rearranging the ***else*** with the ***if*** that follows it.
  - This would be evident if you look at the following code:

```
if ( i == 2 )
      printf (  "With you..." ) ;
else
{
      if ( j == 2 )
      printf ( "...All the time" ) ;
}
```

```
if ( i == 2 )
      printf (  "With you..." ) ;
else if ( j == 2 )
      printf ( "...All the time " ) ;
```

The *Else if Clause*

- Problem
  - A company insures its drivers in the following cases:
    - If the driver is married.
    - If the driver is unmarried, male & above 30 years of age.
    - If the driver is unmarried, female & above 25 years of age.
  - In all other cases the driver is not insured. If the marital status, sex and age of the driver are the inputs.
  - Write a program to determine whether the driver is to be insured or not?

Solution-1

```
/* Insurance of driver - without using logical operators */
main( )
{
      char   sex, ms ;
      int   age ;

      printf ( "Enter age, sex, marital status " ) ;
      scanf ( "%d %c %c", &age, &sex, &ms ) ;

      if ( ms == 'M' )
            printf ( "Driver is insured" ) ;
      else
      {
            if ( sex == 'M' )
            {
                  if ( age > 30 )
                        printf ( "Driver is insured" ) ;
                  else
                        printf ( "Driver is not insured" ) ;
            }
            else
            {
                  if ( age > 25 )
                        printf ( "Driver is insured" ) ;
                  else
                        printf ( "Driver is not insured" ) ;
            }
      }
}
```

Solution-2

```
/* Insurance of driver - using logical operators */
main( )
{
      char   sex, ms ;

      int   age ;

      printf ( "Enter age, sex, marital status " ) ;
      scanf ( "%d %c %c" &age, &sex, &ms ) ;

      if ( ( ms == 'M') || ( ms == 'U' && sex == 'M' && age > 30 ) ||
                        ( ms == 'U' && sex == 'F' && age > 25 ) )
            printf ( "Driver is insured" ) ;
      else
            printf ( "Driver is not insured" ) ;
}
```

Problem…

- Write a program to calculate the salary as per the following table:

Gender	Years of Service	Qualifications	Salary
Male	>= 10	Post-Graduate	15000
	>= 10	Graduate	10000
	< 10	Post-Graduate	10000
	< 10	Graduate	7000
Female	>= 10	Post-Graduate	12000
	>= 10	Graduate	9000
	< 10	Post-Graduate	10000
	< 10	Graduate	6000

Solution

```
main( )
{
      char   g ;
      int   yos, qual, sal ;

      printf ( "Enter Gender, Years of Service and
                  Qualifications ( 0 = G, 1 = PG ):" ) ;
      scanf ( "%c%d%d", &g, &yos, &qual ) ;

      if ( g == 'm' && yos >= 10 && qual == 1 )
            sal = 15000 ;
      else if ( ( g == 'm' && yos >= 10 && qual == 0 ) ||
            ( g == 'm' && yos < 10 && qual == 1 ) )
            sal = 10000 ;
```

Cont…

```
        else if ( g == 'm' && yos < 10 && qual == 0 )
              sal = 7000 ;
        else if ( g == 'f' && yos >= 10 && qual == 1 )
              sal = 12000 ;
        else if ( g == 'f' && yos >= 10 && qual == 0 )
              sal = 9000 ;
        else if ( g == 'f' && yos < 10 && qual == 1 )
              sal = 10000 ;
        else if ( g == 'f' && yos < 10 && qual == 0 )
              sal = 6000 ;

        printf ( "\nSalary of Employee = %d", sal ) ;
}
```

The ! Operator

- This operator reverses the result of the expression it operates on.
  - For example, if the expression evaluates to a non-zero value, then applying ! operator to it results into a 0 and vice versa.
  - If the expression evaluates to zero then on applying ! operator to it makes it 1, a non-zero value. The final result (after applying !) 0 or 1 is considered to be false or true respectively.
    - For Example ! (y < 10), y is less than 10, the ! Operator returns false.

The ! Operator

- The NOT operator is often used to reverse the logical value of a single variable, as in the expression

if (! flag)

Identify the error from the following???

(g)
```
main( )
{
    int x = 30 , y = 40 ;
    if ( x == y )
        printf( "x is equal to y" ) ;
    elseif ( x > y )
        printf( "x is greater than y" ) ;
    elseif ( x < y )
        printf( "x is less than y" ) ;
}
```

(i)
```
main( )
{
    int a, b ;
    scanf ( "%d %d",a, b ) ;
    if ( a > b ) ;
        printf ( "This is a game" ) ;
    else
        printf ( "You have to play it" ) ;
}
```

(h)
```
main( )
{
    int  x = 10 ;
    if ( x >= 2 ) then
        printf ( "\n%d", x ) ;
}
```

Conditional Operator

- The conditional operators ?and :are sometimes called ternary operators since they take three arguments.
  - They form a kind of foreshortened ***if-then-else.***
  - Their general form is,

    expression 1 ? expression 2 : expression 3
  - What this expression says is:
    - "if expression 1 is true (that is, if its value is non-zero), then the value returned will be expression 2, otherwise the value returned will be expression 3".

Examples

- The following program store 3 in y, if x is greater than 5, otherwise it will store 4 in y.

```
(a)   int  x, y ;
      scanf ( "%d", &x ) ;
      y = ( x > 5 ? 3 : 4 ) ;
```

- The comparison with if else is as under,

```
int  x, y ;
scanf ( "%d", &x ) ;
y = ( x > 5 ? 3 : 4 ) ;
```

Equivalent to

```
if ( x > 5 )
    y = 3 ;
else
    y = 4 ;
```

Example

```
char   a ;
int   y ;
scanf ( "%c", &a ) ;
y = ( a >= 65 && a <= 90 ? 1 : 0 ) ;
```

- 1 would be assigned to y ***if a >=65 && a <=90*** evaluates to true, otherwise 0 would be assigned

Important point

- It's not necessary that the conditional operators should be used only in arithmetic statements. This is illustrated in the following examples:

Ex.:
```
int   i ;
scanf ( "%d", &i ) ;
( i == 1 ? printf ( "Amit" ) : printf ( "All and sundry" ) ) ;
```

Ex.:
```
char   a = 'z' ;
printf ( "%c" , ( a >= 'a' ? a : '!' ) ) ;
```

Cont…

- The conditional operators can be nested as shown below.

```
int   big, a, b, c ;
big = ( a > b ? ( a > c ? 3: 4 ) : ( b > c ? 6: 8 ) ) ;
```

- Check out the following conditional expression:

```
a > b ? g = a : g = b ;
```

Limitation

- The limitation of the conditional operators is that after the **?** Or after the **:** only one C statement can occur.
- In practice rarely is this the requirement.
  - Therefore, in serious C programming conditional operators aren't as frequently used as the if-else.


THANKS