Decision Control Structure

Decision Making in C

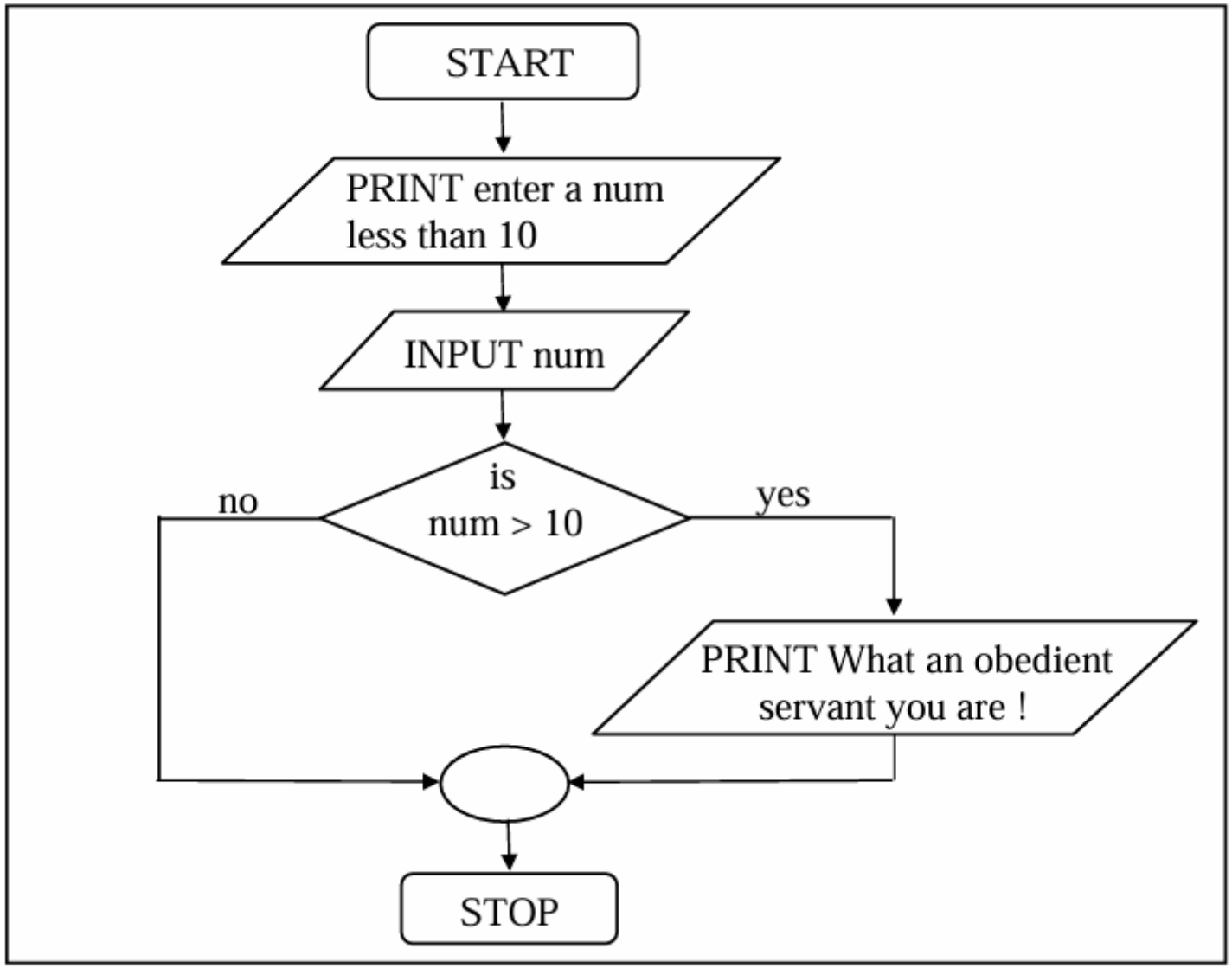
- C has three major decision making instructions—
- the if statement,
- the if-else statement, and
- the switch statement
- A fourth, somewhat less important structure is the one that uses conditional operators
- By default the instructions in a program are executed sequentially.
- However, in serious programming situations, seldom do we want the instructions to be executed sequentially.
- 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

The if Statement

- 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.
- As a general rule, 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.

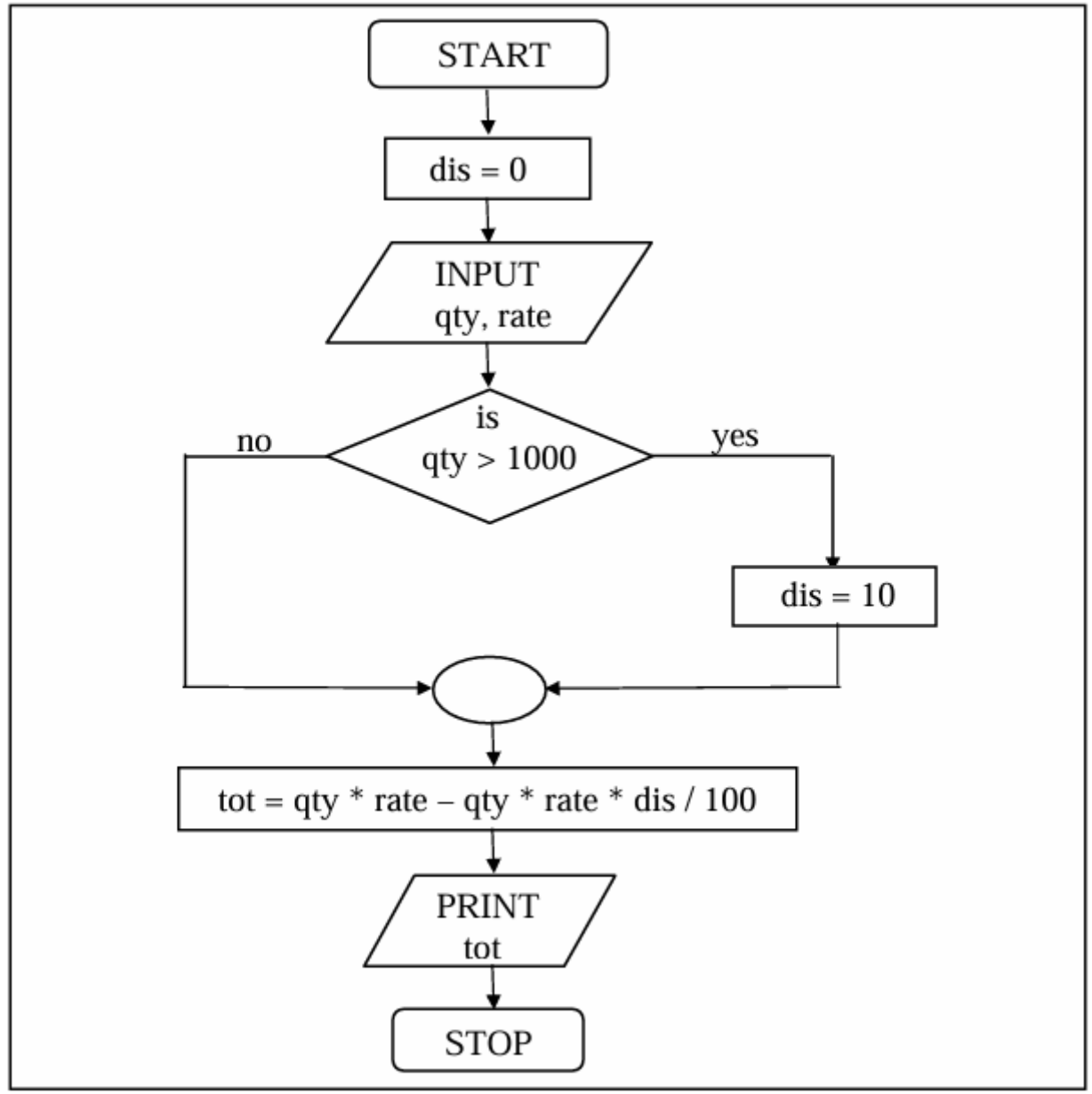
this expression	is true if
x == y	x is equal to y
x != y	x is not equal to y
x < y	x is less than y
x > y	x is greater than y
x <= y	x is less than or equal to y
x >= y	x is greater than or equal to y

```
main( )
{
      int   num ;

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

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


START
PRINT enter a num
less than 10
INPUT num
is
num > 10
no
yes
PRINT What an obedient
servant you are !
STOP

Exercise

- 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.


START
dis = 0
INPUT
qty, rate
is
qty > 1000
no
yes
dis = 10
tot = qty * rate – qty * rate * dis / 100
PRINT
tot
STOP

```
/* 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 ) ;
}
```

The Real Thing

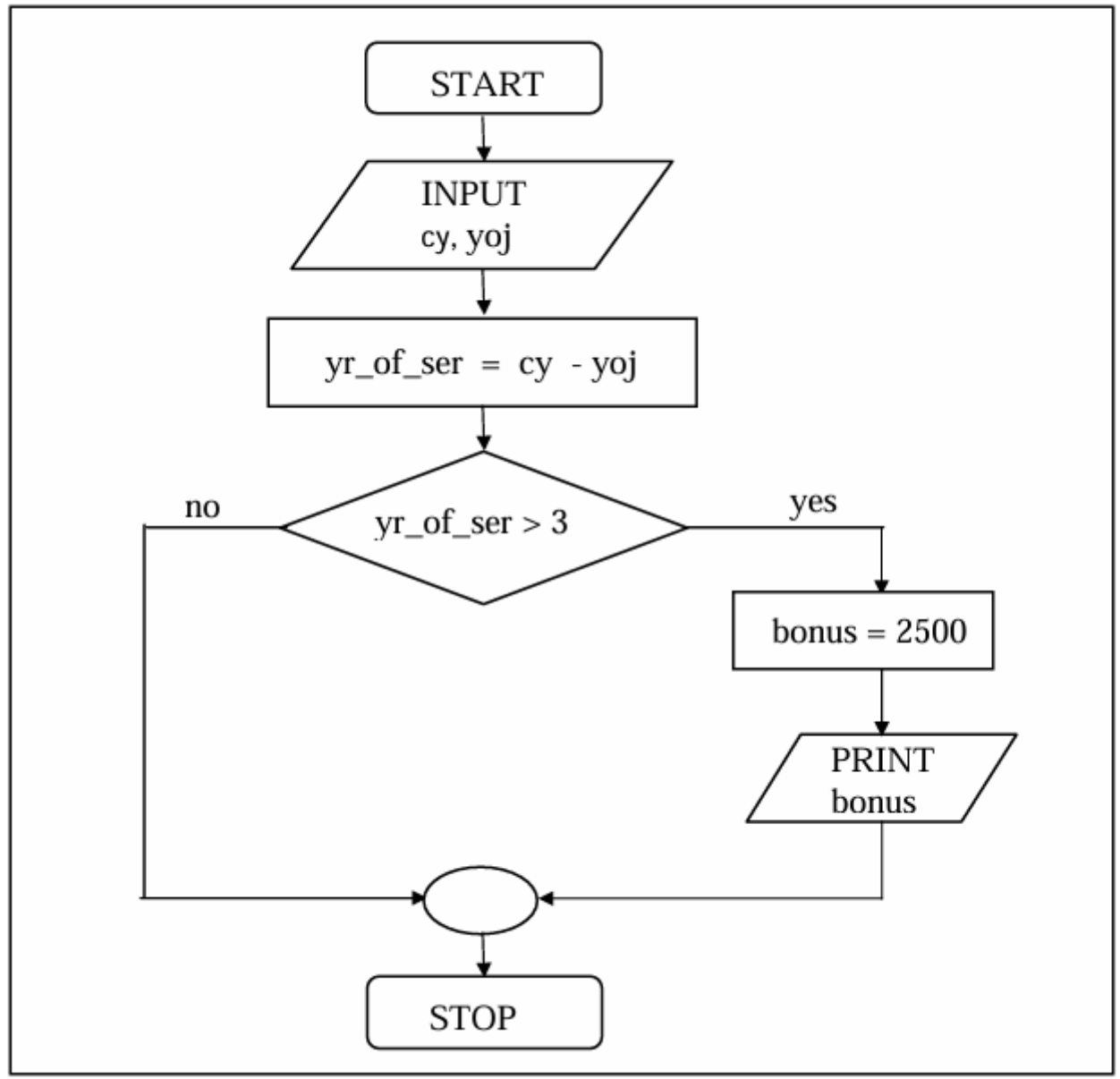
- We mentioned earlier that the general form of the if statement is as follows
- if (condition)
  - statement ;
- Truly speaking the general form is as follows:
- if (expression)
  - statement ;
- Here the expression can be any valid expression including a relational expression. We can even use arithmetic expressions in the if statement
- Note that in C a non-zero value is considered to be true, whereas a 0 is considered to be false

```
if ( 3 + 2 % 5 )
        printf ( "This works" ) ;

if ( a = 10 )
        printf ( "Even this works" )  ;

if ( -5 )
        printf ( "Surprisingly even this works" ) ;
```

Multiple Statements within if

- The current year and the year in which the employee joined the organization are entered through the keyboard. If the number of years for which the employee has served the organization is greater than 3 then a bonus of Rs. 2500/- is given to the employee. If the years of service are not greater than 3, then the program should do nothing.

```
/* 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 ) ;
      }
}
```


START
INPUT
cy, yoj
yr_of_ser = cy - yoj
yr_of_ser > 3
no
yes
bonus = 2500
PRINT
bonus
STOP

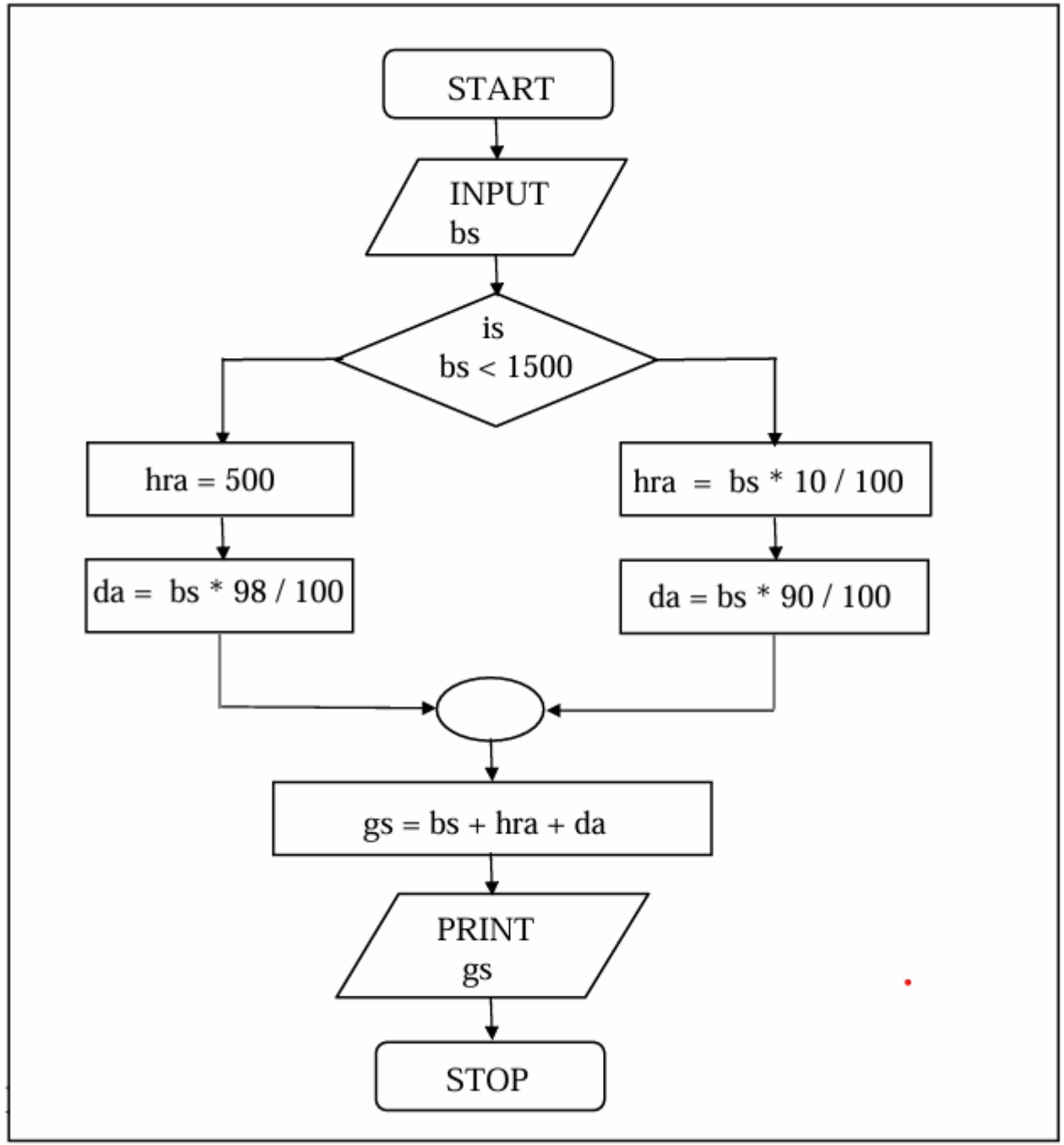
The if-else Statement

- The if statement by itself will execute a single statement, or a group of statements, when the expression following if evaluates to true.
- It does nothing when the expression evaluates to false.
- This is what is the purpose of the else statement
- As with the if statement, the default scope of else is also the statement immediately after the else.
- To override this default scope a pair of braces as shown in the above example must be used.

Exercise

- In a company an employee is paid as under:
- If his basic salary is less than Rs. 1500, then HRA = 10% of basic salary and DA = 90% of basic salary. If his salary is either equal to or above Rs. 1500, then HRA = Rs. 500 and DA = 98% of basic salary. If the employee's salary is input through the keyboard write a program to find his gross salary.

```
/* 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 ) ;
}
```


START
INPUT
bs
is
bs < 1500
hra = 500
da = bs * 98 / 100
hra = bs * 10 / 100
da = bs * 90 / 100
gs = bs + hra + da
PRINT
gs
STOP

Nested if-elses

```
/* A quick demo of nested if-else */
main( )
{
      int   i ;

      printf ( "Enter either 1 or 2 " ) ;
      scanf ( "%d", &i ) ;

      if ( i == 1 )
            printf ( "You would go to heaven !" ) ;
      else
      {
            if ( i == 2 )
                  printf ( "Hell was created with you in mind" ) ;
            else
                  printf ( "How about mother earth !" ) ;
      }
}
```

Important Points

- Each time a if-else construct is nested within another if-else construct, it is also indented to add clarity to the program
- An if-else occurs within the else block of the first if statement.
- Similarly, in some other program an if-else may occur in the if block as well.
- There is no limit on how deeply the ifs and the elses can be nested.

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 !.
- 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, which we would examine later
- The first two operators, && and ||, allow two or more conditions to be combined in an if statement

Exercise

- 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.

```
/* 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" ) ;
            }
      }
}
```

```
/* 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" ) ;
 }
```

Points to Note

- The driver will be insured only if one of the conditions enclosed in parentheses evaluates to true.
- For the second pair of parentheses to evaluate to true, each condition in the parentheses separated by && must evaluate to true.
- Even if one of the conditions in the second parentheses evaluates to false, then the whole of the second parentheses evaluates to false.
- The last two of the above arguments apply to third pair of parentheses as well.

When to apply && and ||

- When it is to be tested whether a value falls within a particular range or not.
- When after testing several conditions the outcome is only one of the two answers (This problem is often called yes/no problem).

The ! Operator

- NOT operator, written as !
- 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

Hierarchy of Operators Revisited

Operators	Type
!	Logical NOT
* / %	Arithmetic and modulus
+ -	Arithmetic
< > <= >=	Relational
== !=	Relational
&&	Logical AND
||	Logical OR
=	Assignment

A Word of Caution

```
main( )
{
      int   i ;

      printf ( "Enter value of i " ) ;
      scanf ( "%d", &i ) ;
      if ( i = 5 )
            printf ( "You entered 5" ) ;
      else
            printf ( "You entered something other than 5" ) ;
}
```

```
main( )
{
      int   i ;

      printf ( "Enter value of i " ) ;
      scanf ( "%d". &i ) :

      if ( i == 5 ) ;
            printf ( "You entered 5" ) ;
}
```

Summary of logical operators

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

The Conditional Operators

- The conditional operators ? and : are sometimes called ternary operators since they take three arguments
- expression 1 ? expression 2 : expression 3
- “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”

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

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

- It’s not necessary that the conditional operators should be used only in arithmetic statements

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

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

- 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 ) ) ;
```

- The limitation of the conditional operators is that after the ? or after the : only one C statement can occur