GOVERNMENT COLLEGE FOR WOMEN(A)

KUMBAKONAM

DEPARTMENT OF COMPUTER SCIENCE

"PROGRAMMING IN C"

STAFF NAME : **S.SUNDARESWARI**

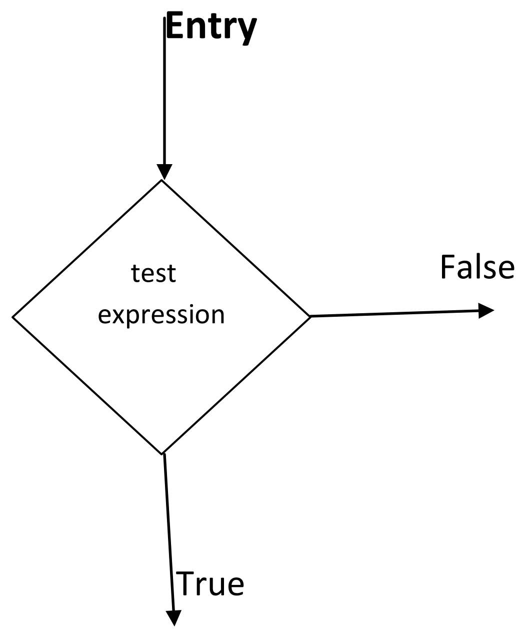
DATE & TIME : **23/09/2020 ,25/09/2020 & 9.30am to 11.30am.**

DECISION MAKING AND BRANCHING STATEMENTS :

1) **if statement**
2) **switch statement**
3) **Conditional operator statement**
4) **goto statement**

If statement is used to control the flow of execution of statements.It is basically a two-way decision statement ,the syntax of if statement is

If (test expression)



Two –way branching

The types of **if** statements are

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

SIMPLE IF STATEMENT

The general form of a simple if statement is

```
If (test expression)
{
    Statement-block;
 }
Statement-x;
```

If the test expression is true, the statement-block will be executed; otherwise the statement-block will be skipped and the execution will jump to the statement-x.

ement-block and the statement-x are executed in se

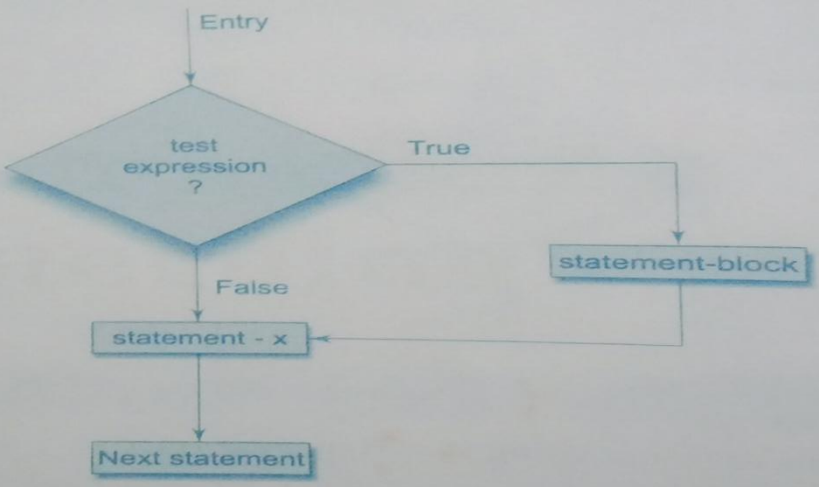


Fig. 5.2 *Flowchart of simple* ***if*** *control*

llowing segment of a program

```
#include<stdio.h>
#include<conio.h>
void main ()
{
  int b;
  printf (" Enter a value");
  scanf ("%d",&b);
   if (b  <  0 )
  printf ("The value is negative");
  getch ();
}
```

<u>THE IF….ELSE STATEMENT :</u>

If the test expression is true ,then the true-block statemet(s),immediately following the if statements are executed; otherwise,the false-block statements are executed.

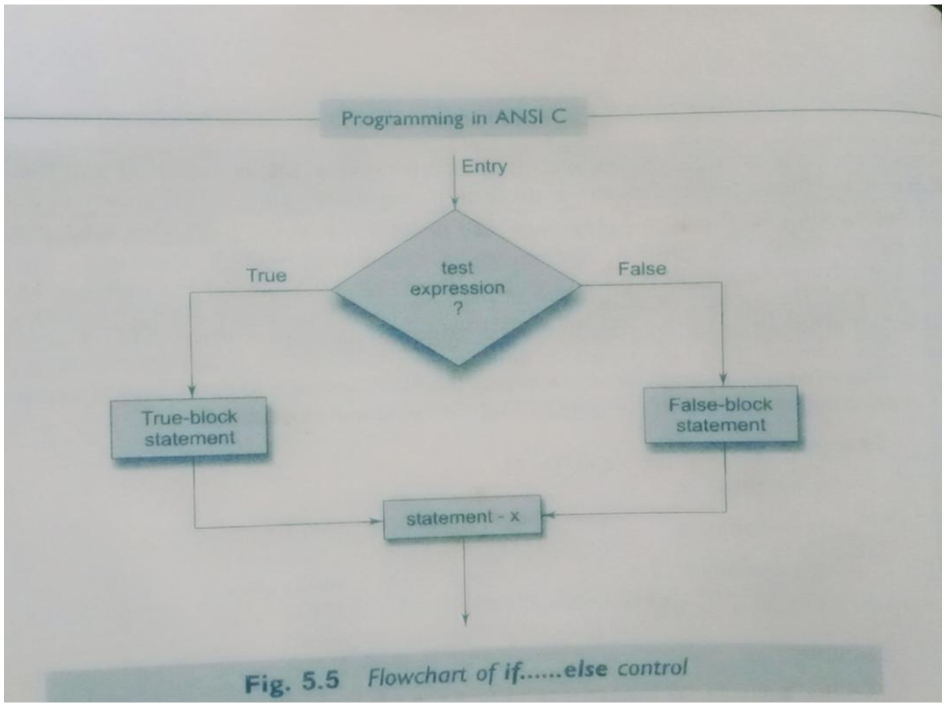


Fig. 5.5 *Flowchart of* ***if......else*** *control*

The general form is

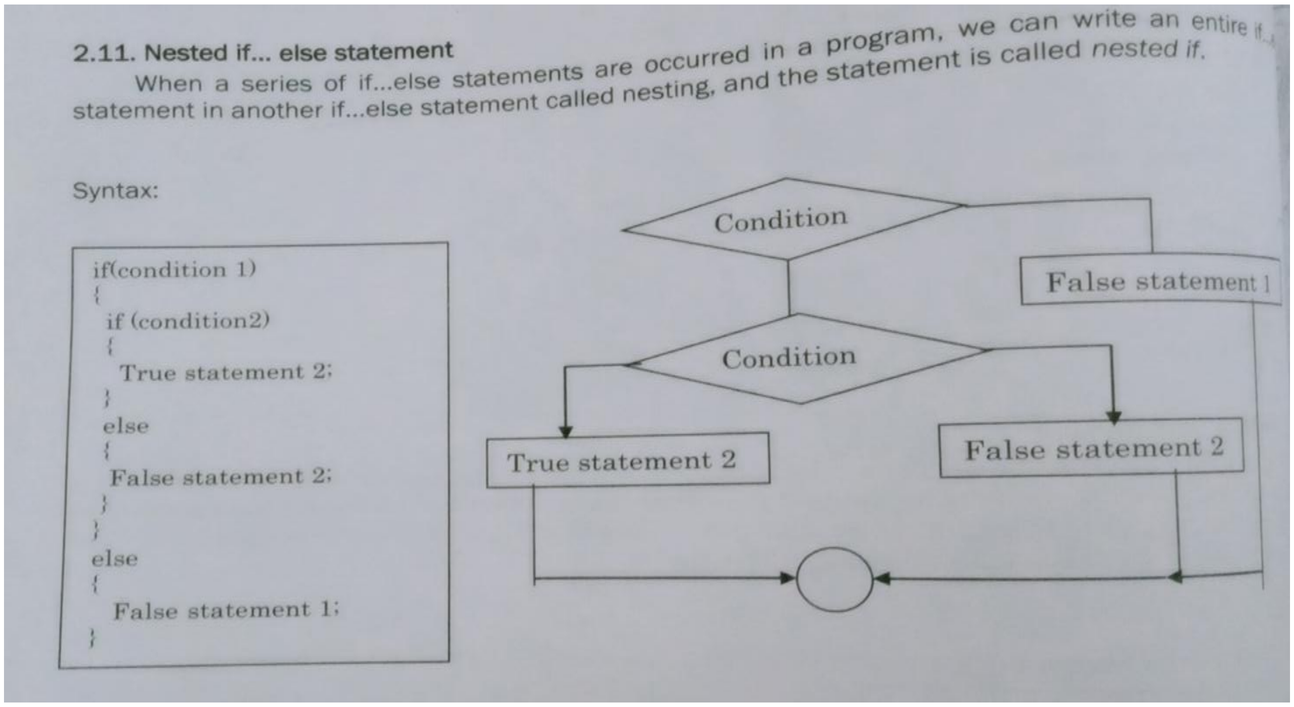
```
    If ( test expression)
{
          True-block statement (s);
}
else
{
        False-block statement (s);
}
Statement-x
```

Example :

```
#include <stdio.h>

#include<conio.h>

void main ()

{

  Int b;

 printf (“Enter a value :”);

 scanf (“%d”, &b);

if (b < 0)

        printf (“The value is negative\n”);

else if (b==0)

        printf (“The value is zero\n”);

  else

       printf (“The value is positive\n”);

getch ();

}
```

NESTING OF IF …ELSE STATEMENTS :

If the condition -1 is false, the statement -3 will be executed; otherwise it continues to perform the second test.If the condition -2 is true, the statement -1 will be evaluated; otherwise the statement-2 will be evaluated and then the control is transferred to the statement-x.



2.11. Nested if... else statement

When a series of if...else statements are occurred in a program, we can write an entire if... statement in another if...else statement called nesting, and the statement is called *nested if*.

Syntax:

```
if(condition 1)
{
 if (condition2)
 {
  True statement 2;
 }
 else
 {
  False statement 2;
 }
}
else
{
  False statement 1;
}
```

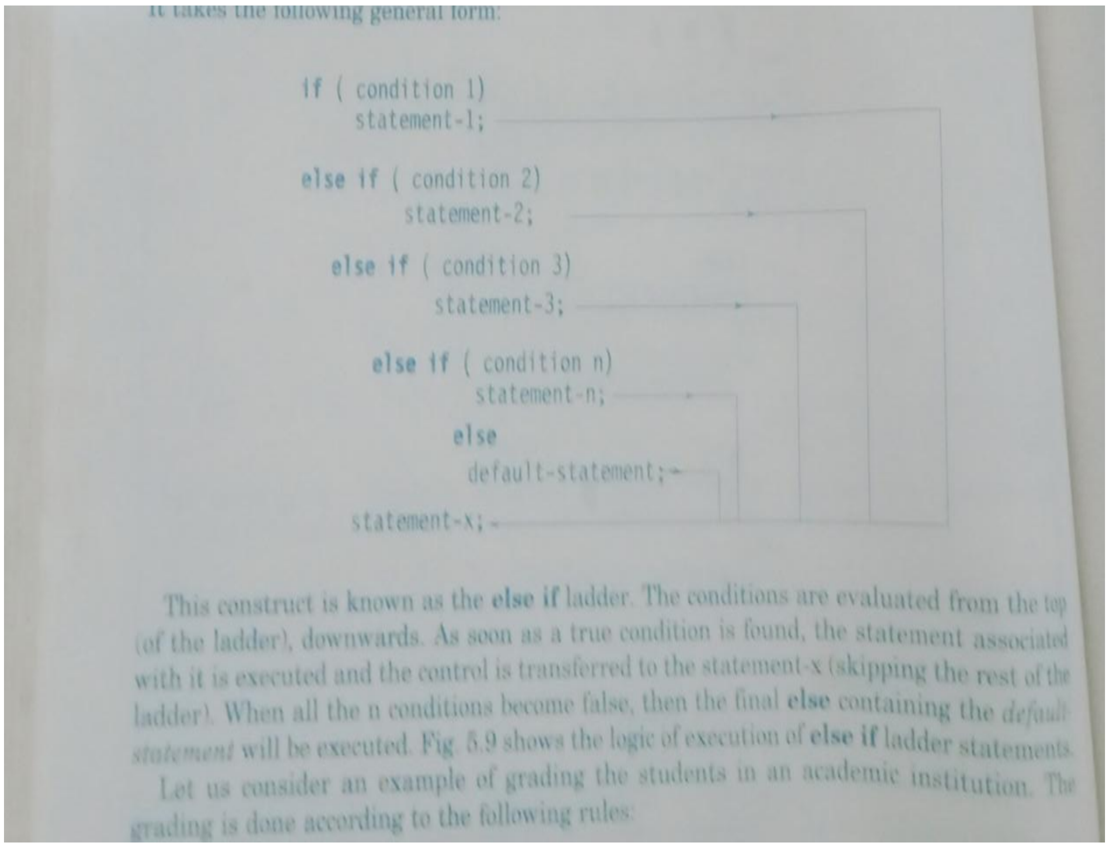
Condition
False statement 1
Condition
True statement 2
False statement 2

#include<stdio.h>

#include<conio.h>

```
void main()
{
   Int n;
  printf (“ Enter any number”);
  scanf(“%d”,&n);
  if (n == 11)
    printf (“Play cricket match”);
else
{
    If (n==6)
          printf (“Play Ball Badminton”);
else
          printf (“ Play any game”);
}
}
```

THE ELSE IF LADDER



It takes the following general form:

```
if ( condition 1)
    statement-1;

else if ( condition 2)
        statement-2;

    else if ( condition 3)
            statement-3;

         else if ( condition n)
                 statement-n;
              else
                 default-statement;

statement-x;
```

This construct is known as the **else if** ladder. The conditions are evaluated from the top (of the ladder), downwards. As soon as a true condition is found, the statement associated with it is executed and the control is transferred to the statement-x (skipping the rest of the ladder). When all the n conditions become false, then the final **else** containing the *default statement* will be executed. Fig. 5.9 shows the logic of execution of **else if** ladder statements.

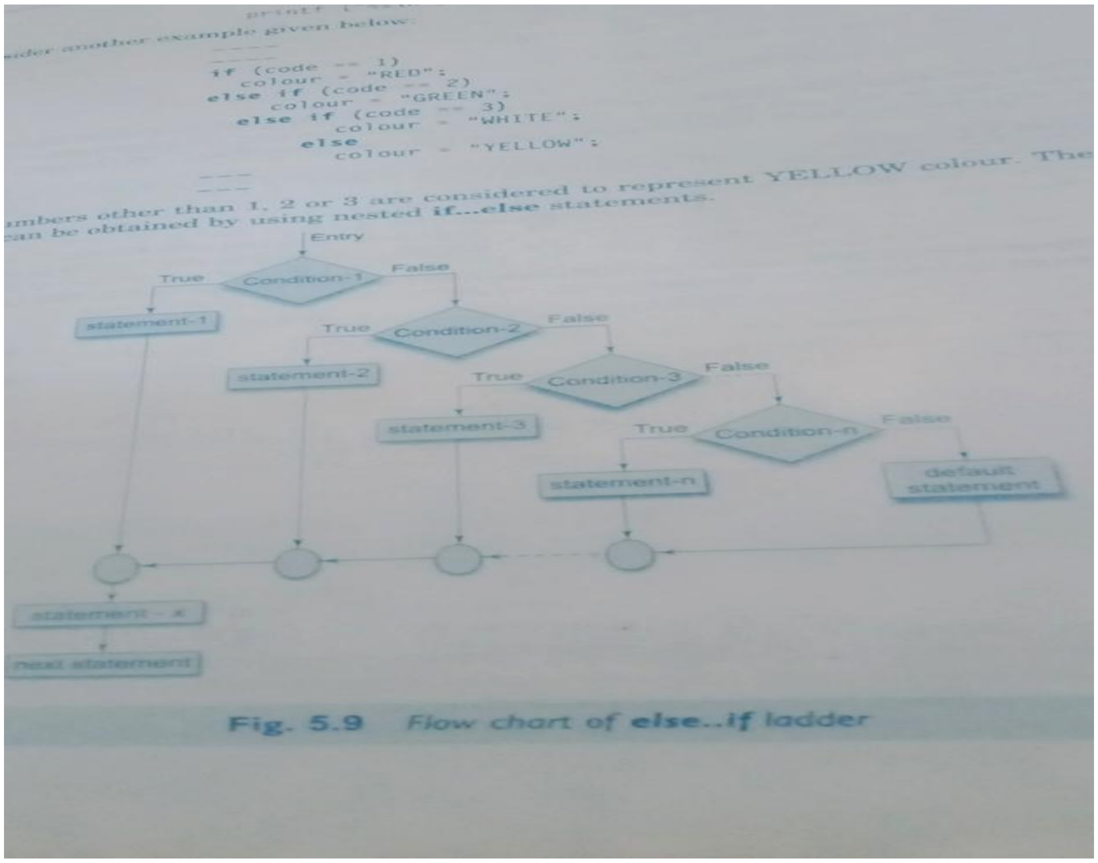
Let us consider an example of grading the students in an academic institution. The grading is done according to the following rules:

with it is executed and the control is transferred to the statem
ladder). When all the n conditions become false, then the final **else** conta
statement will be executed. Fig. 5.9 shows the logic of execution of **else if** l
Let us consider an example of grading the students in an academi
grading is done according to the following rules:

Average marks	**Grade**
80 to 100	Honours
60 to 79	First Division
50 to 59	Second Division
40 to 49	Third Division
0 to 39	Fail

This grading can be done using the **else if** ladder as follows:

```
if (marks > 79)
    grade = "Honours";
  else if (marks > 59)
      grade = "First Division";
    else if (marks > 49)
        grade = "Second Division";
      else if (marks > 39)
              grade = "Third Division";
```



printf (

ider another example given below:

```
-----
-----
if (code == 1)
    colour = "RED";
else if (code == 2)
        colour = "GREEN";
     else if (code == 3)
             colour = "WHITE";
          else
             colour = "YELLOW";
---
---
```

mbers other than 1, 2 or 3 are considered to represent YELLOW colour. The
an be obtained by using nested **if...else** statements.

Fig. 5.9 *Flow chart of* ***else..if*** *ladder*

SWITCH STATEMENT :

The switch statement is used to pickup or executes a particular group of statement from several available groups of statements.It allows us to make decision from the number of choices.It is a multi-way statement.

The general syntax form

```
switch (expression)
{
   case   value-1 :
```

```
                    block-1
                     break;
  case   value-2  :
                   block-2
                   break;
……
default   :
             default-block
              break;
}
Statement-x;
```

The expression is an integer expression or characters.Value -1,Value-2 ,Value -3 are constant expressions and are known as case labels.Block- 1,Block-2 are statement lists and may contain zero or more statements.

When the switch is executed, the value of the expression is successfully compared against the values value-1, value-2 .If a case is found whose value matches with the value of the expression,then the block of statements that follows the case are executed.

The break statement at the end of wach block signals the end of a particular case and causes an exit form the switch statement,transferring the control of the statement-x following the switch.

The default is an optional case.When present ,it will be executed if the valueof the expression does not match with any of the case values.If not present,no action takes place if all matches fail and the control goes to the statement-x.

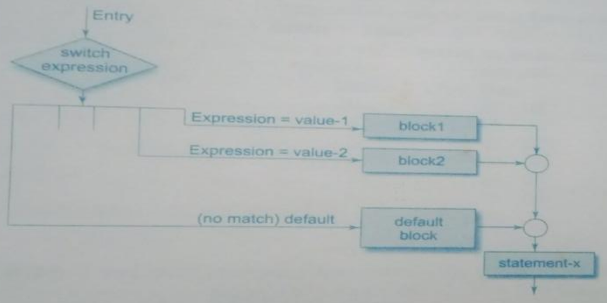
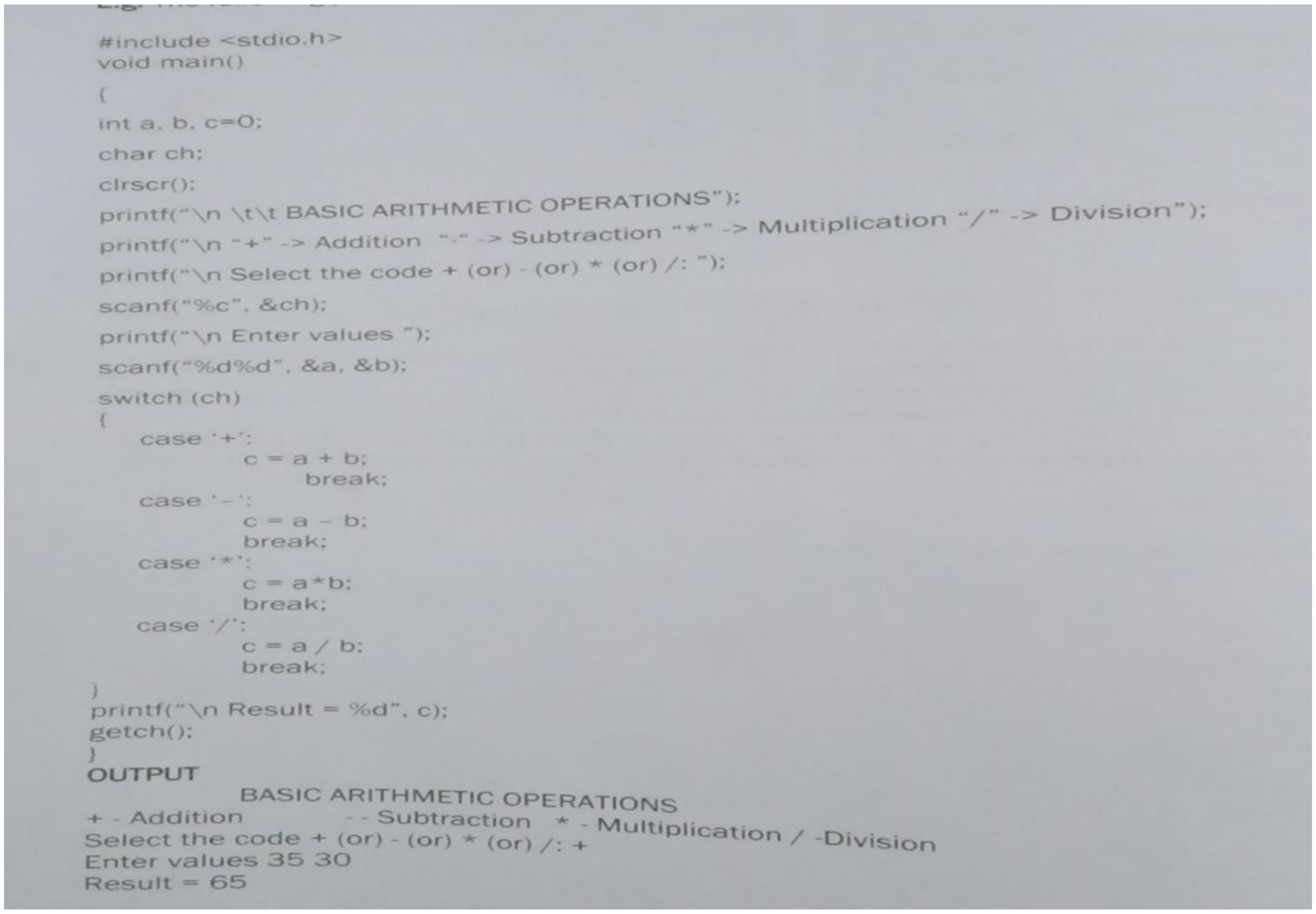


Fig. 5.11 *Selection process of the **switch** statement*



```
#include <stdio.h>
void main()
{
int a, b, c=0;
char ch;
clrscr();
printf("\n \t\t BASIC ARITHMETIC OPERATIONS");
printf("\n "+" -> Addition  "-" -> Subtraction "*" -> Multiplication "/" -> Division");
printf("\n Select the code + (or) - (or) * (or) /: ");
scanf("%c", &ch);
printf("\n Enter values ");
scanf("%d%d", &a, &b);
switch (ch)
{
    case '+':
            c = a + b;
                break;
    case '-':
            c = a - b;
            break;
    case '*':
            c = a*b;
            break;
    case '/':
            c = a / b;
            break;
}
printf("\n Result = %d", c);
getch();
}
```

OUTPUT

```
             BASIC ARITHMETIC OPERATIONS
+ - Addition          - - Subtraction  * - Multiplication / -Division
Select the code + (or) - (or) * (or) /: +
Enter values 35 30
Result = 65
```

THE CONDITIONAL OPERATOR ?: STATEMENT

This operator is a combination of ? and :, and takes three operands.This operator is popularly known as "conditional operator".

Conditional expression ? expression 1 : expression 2

The conditional expression is evaluated first. If the result is nonzero, expression 1 is evaluated and is returned as the value of the conditional expression.Otherwise,expression2 is evaluated and its value is returned.

```
 If (x  <  0)
              flag  = 0;
else
               flag = 1;
```

can be written as

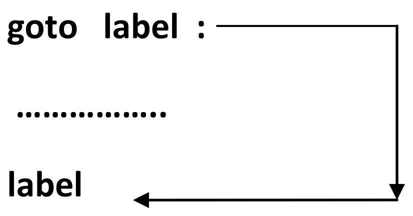
```
   flag =  (x  < 0  )  ?  0  : 1
```

__THE   GOTO   STATEMENT__

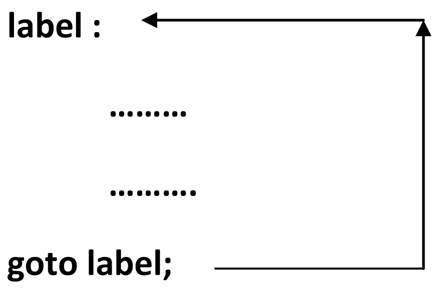
_“C” provides the goto statement to transfer control unconditionally from one place to another place in the program. A goto statement can cause program control to end up almost anywhere in the program unconditionally.

A goto statement requires a label to identity the place to move the execution. A label is a valid variable name and must be ended with colon.

The syntax for goto statement is ,



```
goto   label  :
 ……………..
label
```



```
label :
          ………
          ……….
goto label;
```

```
#include <stdio.h>
#include<conio.h>
void main()
{
   Int a,b;
   clrscr();
```

```
  printf (“ Enter the two numbers a and b”);

  scanf (“%d %d”, &a,&b);

  if (a==b)

   goto equal;

  else

     printf (“A is not equal to B”);

  exit (0);

      equal : printf (“A is equal to  B”);

}
```