Control Statements

Acknowledgement

The contents (figures, concepts, graphics, texts etc.) of the slides are gathered and utilized from the books mentioned and the corresponding PPTs available online:

Books:

1. **Let Us C,** Yashawant Kanetkar, BPB Publications.
2. **The C Programming Language,** B. W. Kernighan, D. Ritchie, Pearson Education India.

Web References:

1. **Problem Solving through Programming in C,** Anupam Basu, NPTEL Video Lectures. Link: https://nptel.ac.in/courses/106/105/106105171/
2. **Compile and Execute C Online (Link:** https://www.onlinegdb.com/)

Chapter Objectives

- To learn about different types of Control Statements

Chapter Topics

- Decision Structures – if statement, if-else statement, nested if statement
- The switch statement
- Repetition or Iteration structure - for statement, continue statement, nested loop, while loop

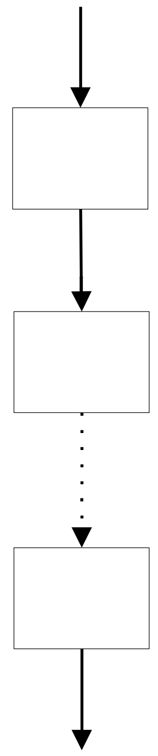
Decision making and Branching

- To change the order of execution based on certain conditions or repeat a group a statements until certain specified conditions are met.
- Any expression can be used as a program statement by following the expression with a ';' (semicolon).
- ANSI C has the following categories of statements
  - Selection – if, switch
  - Iteration – for, do, while
  - Jump – continue, break, goto, return
  - Label – case, default, (goto) label statement
  - Expression – valid expression
  - Block – { … } (also called compound statements)

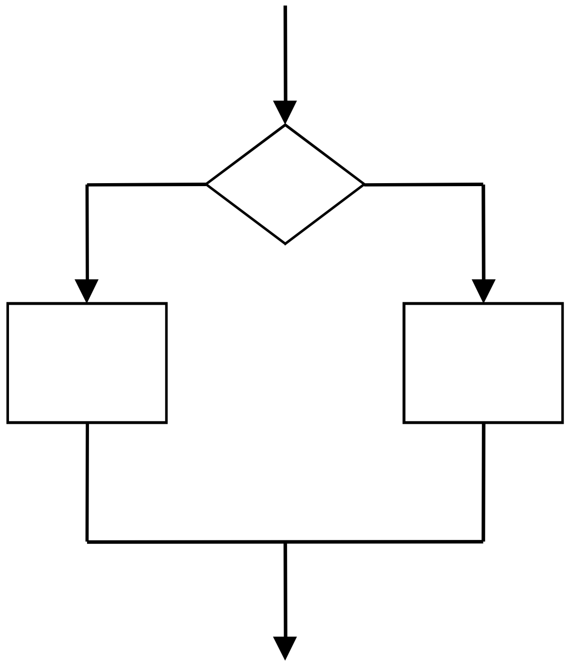
C Control Structure Decision

Compound Statements

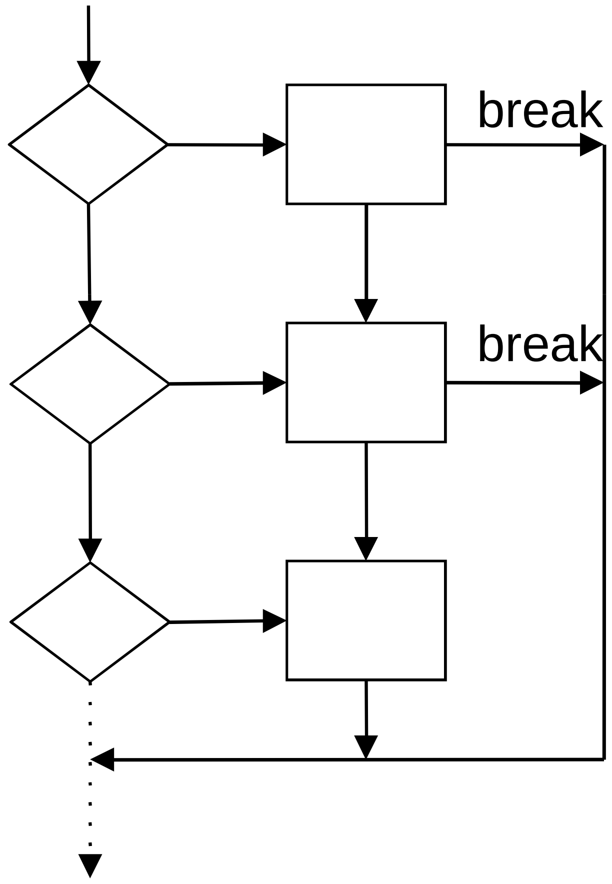
{ . . . }



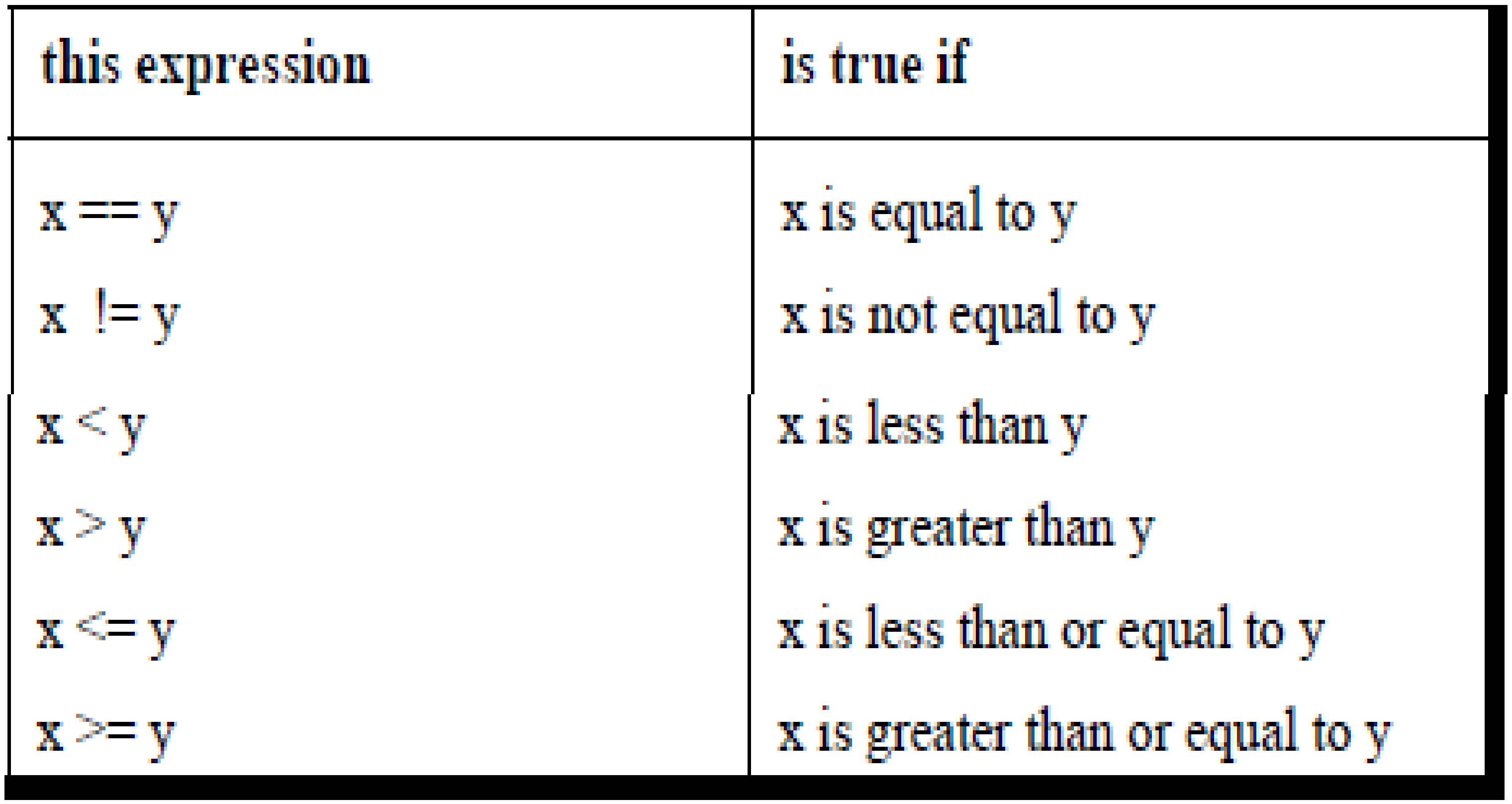
if - else



switch - case



Arithmetic, Relational and Logical Operator



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

Example: if condition

```
    if ( this condition is true )
            execute this statement ;
Simply:
            if(condition)
            statement;

Or
            if(condition)
                    {
                    Statement 1;
                    Statement 2;
                    ….
                    Statement n;
                    }
```

The if – else Statement

- Structure *if (expression)*

  statement_1

  else

- The *else* part is optional
- The expression is evaluated: if *expression* is *TRUE* (I.e. non zero) then *statement_1*. If *expression* is *FALSE* (i.e. zero) then *statement_1* is executed if present. For multiple *if*'s, the else goes with the closest *if* without an *else* condition.

Contd…

```
main()
{
int x=0;
if(x==0)
printf("X is zero");
}
```

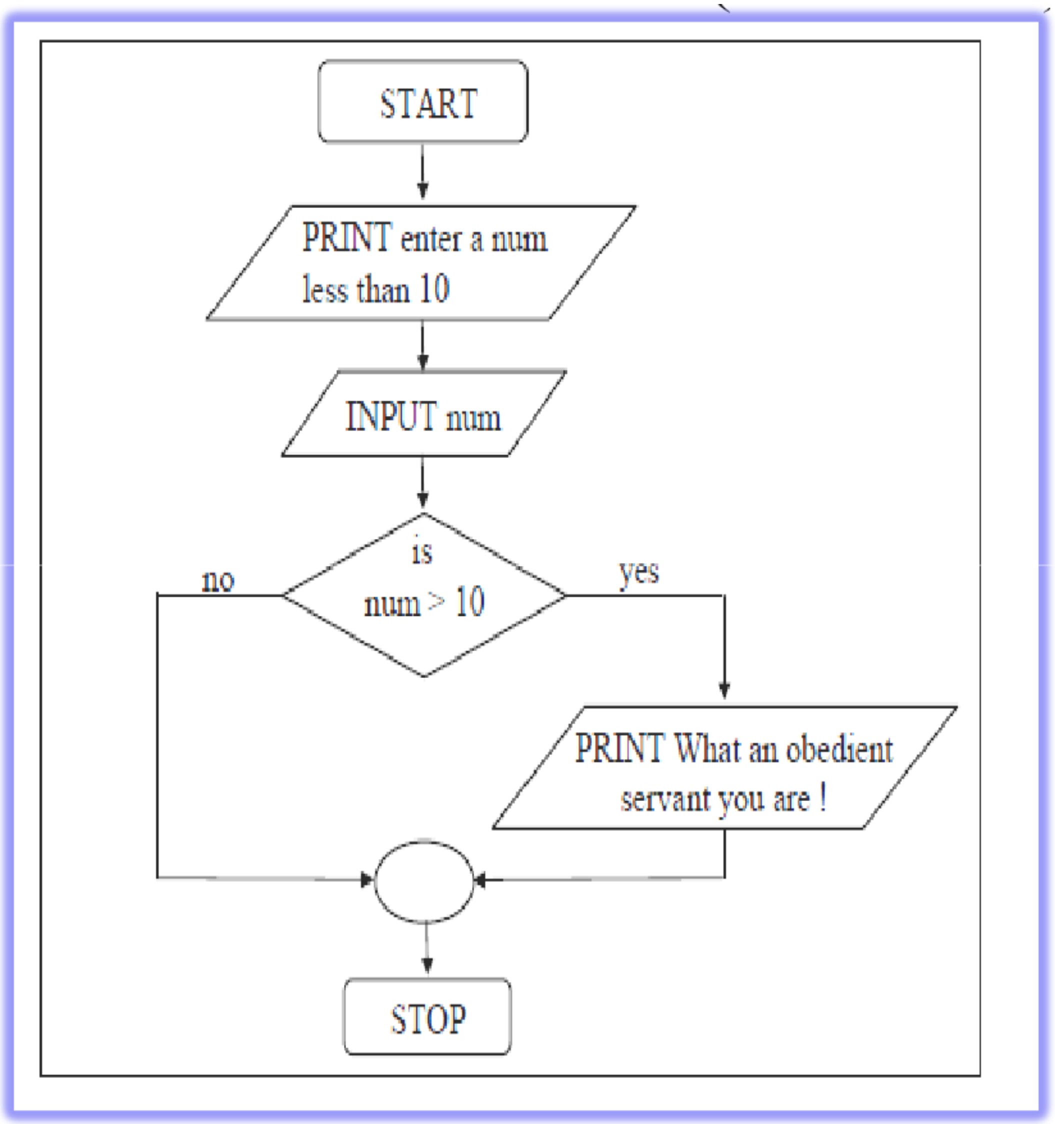
All control statements can be used in conjunction with relational and logical operator.

```
main()
{
int x=0,y;
if(x>0)
{
printf("X is positive");
y=x+3;
printf("%d",y);
}
}
```

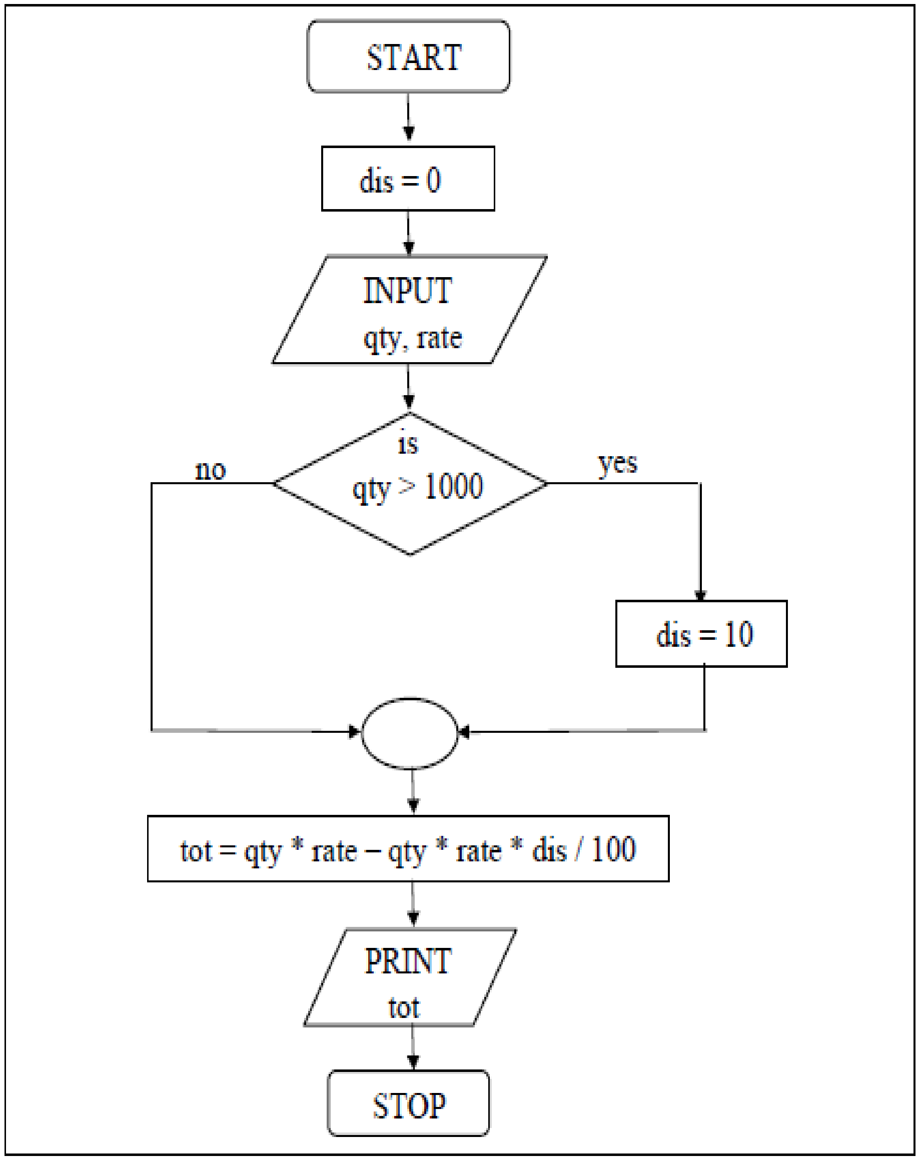
contd..

```
/* program to check greater between two numbers */
main()
{
int a,b;
printf(“Enter two numbers”);
scanf(“%d %d”,&a,&b);
if(a>b)
printf(“%d is greater than %d”,a,b);
}
```

if (contd…)

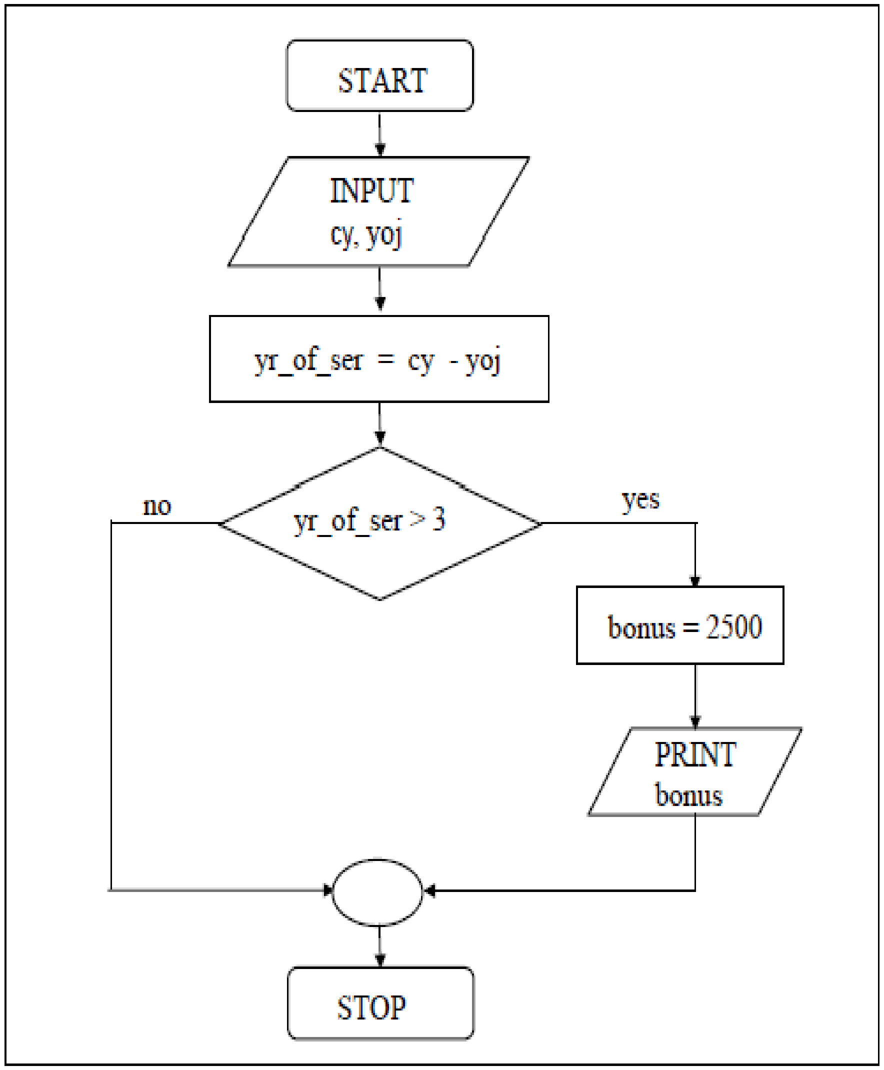

START
PRINT enter a num less than 10
INPUT num
is num > 10
no
yes
PRINT What an obedient servant you are !
STOP

```
main()
{
int num;
printf("Enter a number")
scanf("%d",&num);
if(num>10)
printf("What an obedient
servant you are");
}
```



```
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 ) ;
}
```

Example : Compound Statement within if



```
/* 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 ) ;
 }
else
 {
   }
}
```

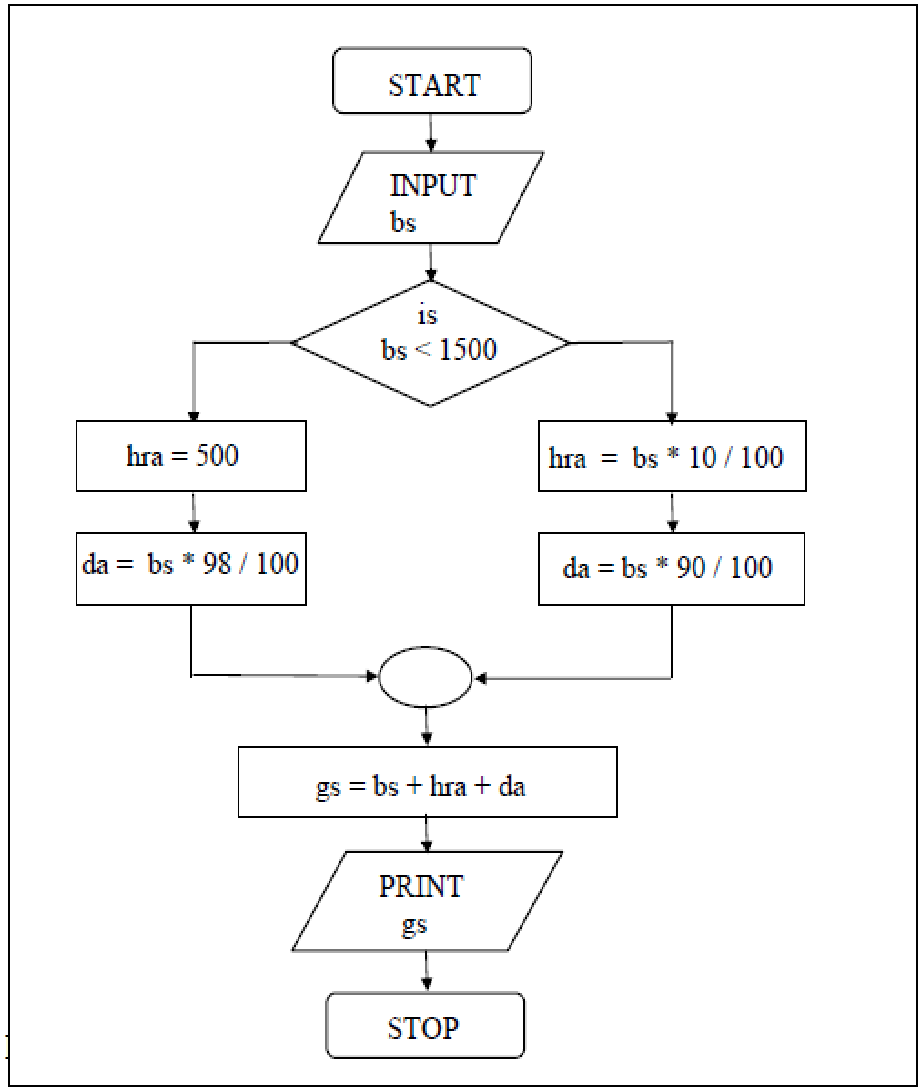
if-else

- 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.
- We can execute one group of statements if the expression evaluates to true and another group of statements if the expression evaluates to false.
- That is the purpose of the else statement that is demonstrated in the following example:

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.

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 )              ;
}
```



- The group of statements after the if upto and not including the else is called an ‘if block’. Similarly, the statements after the else form the ‘else block’.
- Notice that the else is written exactly below the if. The statements in the if block and those in the else block have been indented to the right. This formatting convention is followed throughout the book to enable you to understand the working of the program better.
- Had there been only one statement to be executed in the if block and only one statement in the else block we could have dropped the pair of braces.
- 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.

if-else

if-else is used to make decision based upon certain condition

```
if (expression)
statement 1;
else
statement 2;
```

```
if (a>0)
{
printf("a is positive");
}
else
printf("a is negative");
```

```
if (expression)
{
statement 1;
statement 2;
…
statement n;
}
else
{
Statement k;
Statement z;
…
Statement L;
}
```

```
if (a>0)
{
printf("a is positive");
z=a+b;
}
else
{
printf("a is negative");
z = a;
}
```

z =a+b, a>0
z =a   , otherwise

Nested *if-elses*

- It is perfectly all right if we write an entire if-else construct within either the body of the if statement or the body of an else statement.
- This is called ‘nesting’ of ifs. This is shown in the following program.

```
/* A quick demo of nested if-else
*/  main( )
{
int i ;
printf ( "Enter either 1 or 2 " ) ;  scanf (
"%d", &i ) ;

if ( i == 1 )
  {
    printf ( "You are male !" ) ;
   }
else
 {
   if ( i == 2 )
        printf ( “You are female");
   else
         printf ( “Gender not verified !" ) ;
 }
}
```

- Note that the second if-else construct is nested in the first else statement. If the condition in the first if statement is false, then the condition in the second if statement is checked. If it is false as well, then the final else statement is executed.
- You can see in the program how each time a if-else construct is nested within another if-else construct, it is also indented to add clarity to the program.

- In the above 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 ;
          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 ;
```

if-else with logical operator

- C allows usage of three logical operators, namely, &&, || and !. These are to be read as ‘AND’ ‘OR’ and ‘NOT’ respectively.
- The first two operators, && and ||, allow two or more conditions to be combined in an if statement.

Consider the following example.

Example 2.4: 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.

There are two ways in which we can write a program for this example. These methods are given below.

if-else with logical operator

```
/* Method – I */
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" ) ;
            }
        }
}
```

if-else with logical operator

This is a straight forward program. Observe that the program usesnested if-elses. This leads to three disadvantages:

(a) As the number of conditions go on increasing the level of indentation also goes on increasing.

(b) As a result the whole program creeps to the right. Care needs to be exercised to match the corresponding ifs and elses.

(c) Care needs to be exercised to match the corresponding pair of braces.

All these three problems can be eliminated by usage of ‘Logical operators’

.All these three problems can be eliminated by usage of ‘Logical operators’. The following program illustrates this.

```
/* Method – II */
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" ) ;
}
```

As can be seen from the second if statement, the && operator is used to combine two conditions. ‘Second division’ gets printed if both the conditions evaluate to true. If one of the conditions evaluate to false then the whole thing is treated as false.

if-else with logical operator

Two distinct advantages can be cited in favour of this program:

(a) The matching (or do I say mismatching) of the ifs with their corresponding elses gets avoided, since there are no elses in this program.

(b) In spite of using several conditions, the program doesn't creep to the right. In the previous program the statements went on creeping to the right. This effect becomes more pronounced as the number of conditions go on increasing. This would make the task of matching the ifs with their corresponding elses and matching of opening and closing braces that much more difficult.

if-else with logical operator

```
 main()
{
int x=3,y=9,z;
if(x>=3 && y == 9)
z = x+y;
else
z=x+9;
}
```

```
 main()
{
int x=3,y=9,z;
if(x>=3 || y == 9)
z = x+y;
else
z=x+9;
}
```

```
 main()
{
int x=3,y=9,z=5,k;
if(x>=3 || y == 9 && z < 5)
z = x+y;
else
z=x+9;
}
```

/* else if ladder demo */

```
main( )
{
int m1, m2, m3, m4, m5, per ;
per = ( m1+ m2 + m3 + m4+ m5 ) / per ;
if ( per >= 60 )
        printf ( "First division" ) ;
else if ( per >= 50 )
        printf ( "Second division" ) ;
else if ( per >= 40 )
        printf ( "Third division" ) ;
else
        printf ( "fail" ) ;
}
```

You can note that this program reduces the indentation of the statements. In this case every else is associated with its previous if. The last else goes to work only if all the conditions fail. Even in else if ladder the last else is optional

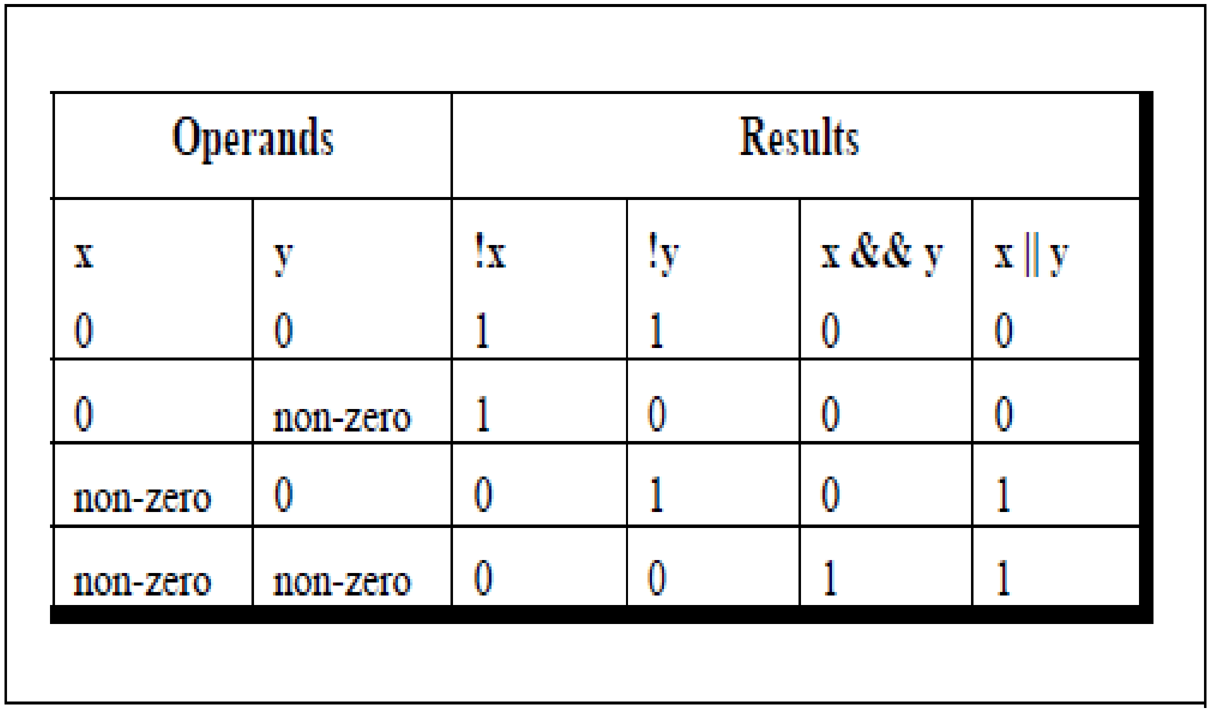
Problem

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.



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

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" ) ;
       }
  }
}
```

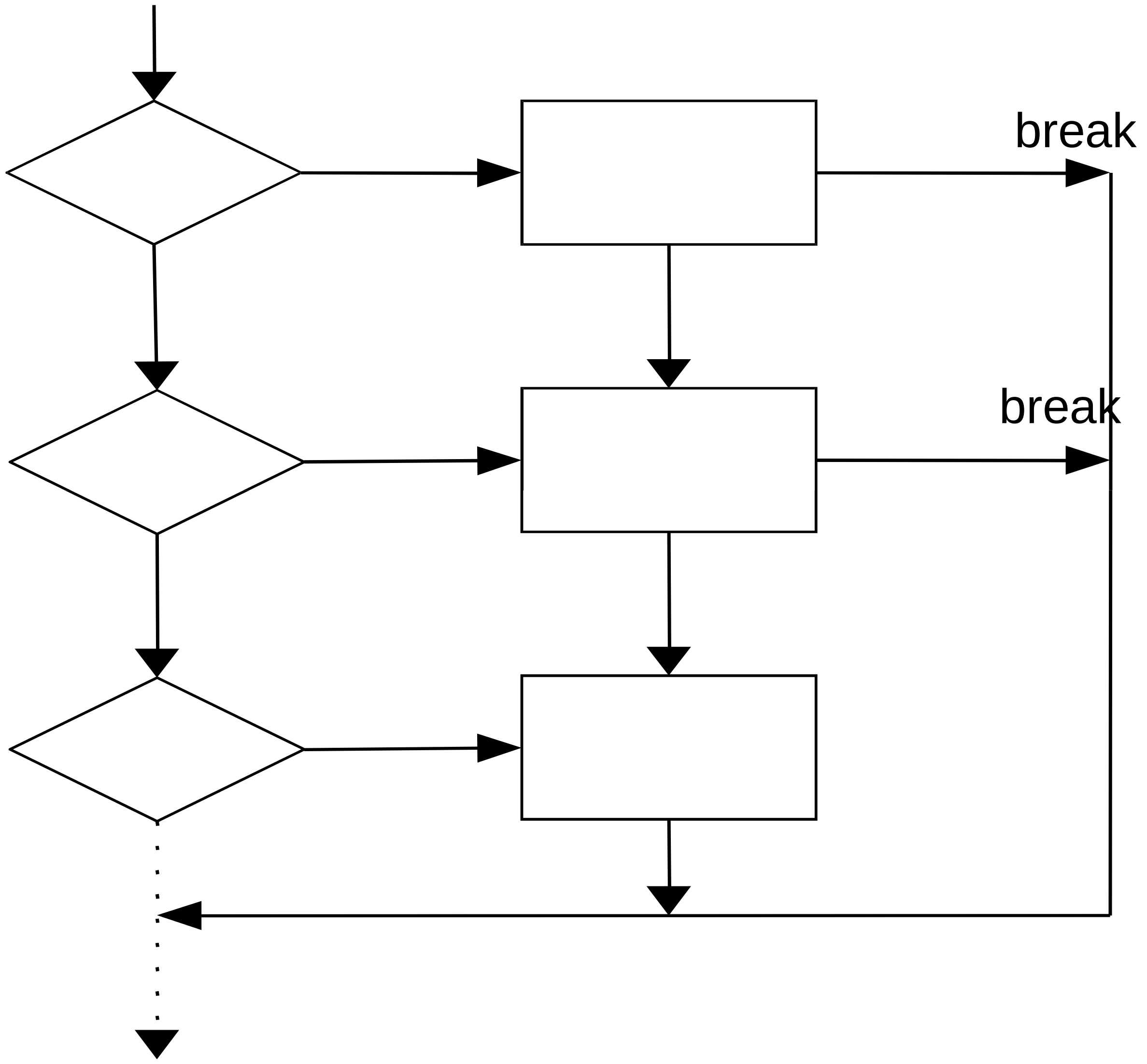
```
/* Insurance of driver - using logical
operators */
main( )
{
char sex, ms ;  int age ;
printf ( "Enter age, sex, marital status " ) ;
scanf ( "%d %c %c" &age, &sex, &ms ) ;

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" ) ;
}
```

The switch-case Statement

- –AKA switch-case-default
- –Multiple branch selection statement
- –Tests the value of an expression against a list of integer or char constants
- –When a match is found, then statement associated with that constant is executed.
- –Structure

```
switch (expression)
{
        case I1: statements;
        case I2: statements;
        case I3: statements;
        case In: statements;
        default: statements; // optional
}
```

switch - case

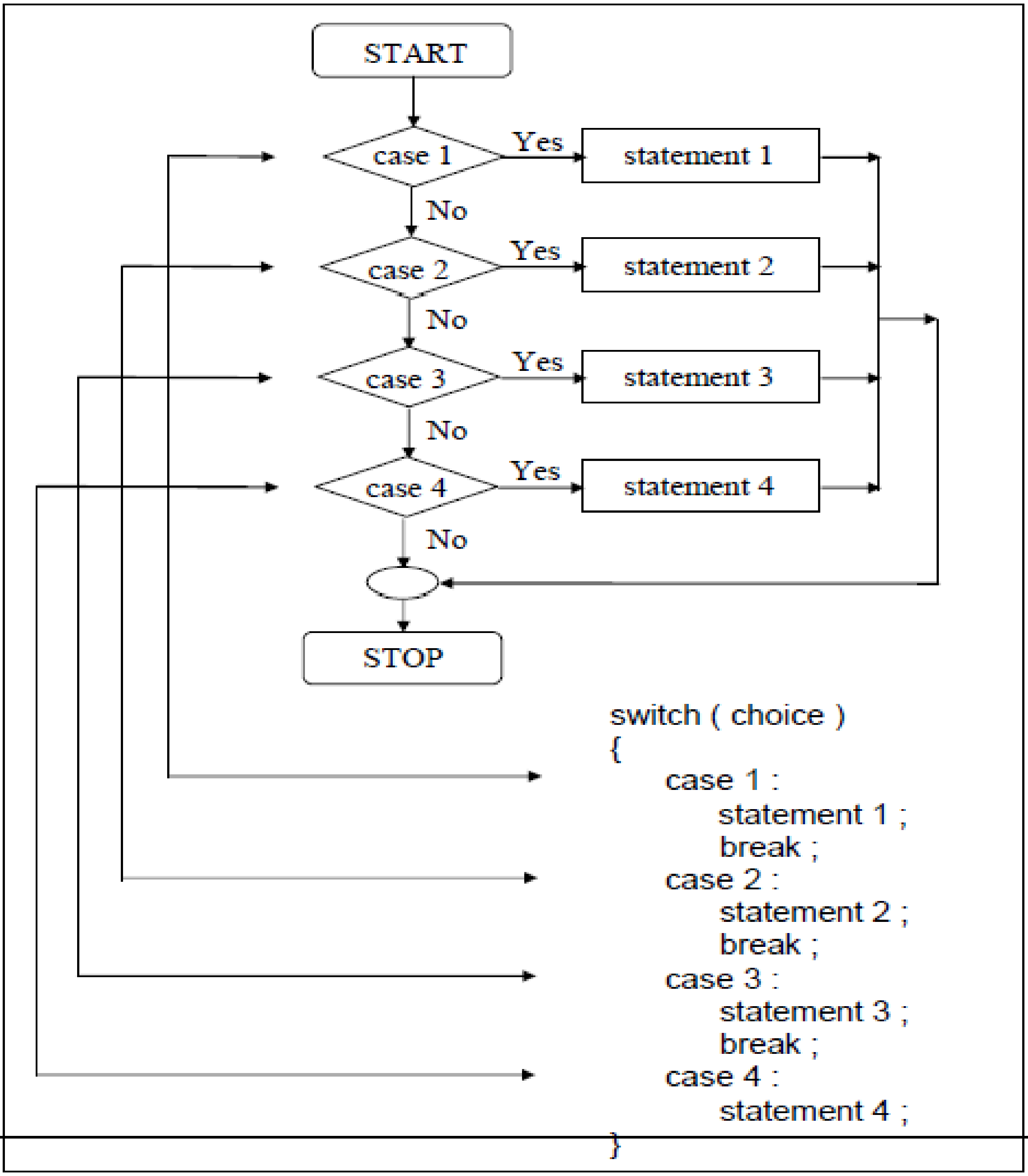


The switch-case Statement contd…

- Operation: the expression is evaluated; then execution continues at the statements following the case statement that matches the result or after the label *default* if there are no matches. If the *default* case does not exist, then execution continues after the last case statement.
- Execution continues through remaining cases in the switch structure unless the *break* instruction is encountered. If a *break* is encountered, then execution continues after the present *switch-case* instance.

```
switch ( integer expression )
{
case constant 1 :
do this ;
case constant 2 :
do this ;
case constant 3 :
do this ;
default :
do this ;
}
```

```
main( )
{
int i = 2 ;
switch ( i )
{
case 1 :
printf ( "I am in case 1 \n" ) ;
break;
case 2 :
printf ( "I am in case 2 \n" ) ;
break;
case 3 :
printf ( "I am in case 3 \n" ) ;
break;
default :
printf ( "I am in default \n" ) ;
}
}
```



```
switch ( choice )
{
     case 1 :
          statement 1 ;
          break ;
     case 2 :
          statement 2 ;
          break ;
     case 3 :
          statement 3 ;
          break ;
     case 4 :
          statement 4 ;
}
```

```
main( )
{
int i = 22 ;
switch ( i )
{
case 121 :
printf ( "I am in case 121 \n" ) ;
break ;
case 7 :
printf ( "I am in case 7 \n" ) ;
break ;
case 22 :
printf ( "I am in case 22 \n" ) ;
break ;
default :
printf ( "I am in default \n" ) ;
}
}
```

```
main( )
{
char c = 'x' ;
switch ( c )
{
case 'v' :
printf ( "I am in case v \n" ) ;
break ;
case 'a' :
printf ( "I am in case a \n" ) ;
break ;
case 'x' :
printf ( "I am in case x \n" ) ;
break ;
default :
printf ( "I am in default \n" ) ;
}
}
```

```
switch ( ch )
{
case 'a' :
case 'A' :
printf ( "a for apple" ) ;
break ;
case 'b' :
case 'B' :
printf ( "b for ball" ) ;
break ;
case 'c' :
case 'C' :
printf ( "c for cat" ) ;
break ;
default :
printf ( “Capital A,B or C and small a, b, c is
not stored in ch" ) ;
}
}
```

Valid Switch Expression

```
switch ( i + j * k )

switch ( 23 + 45 % 4 * k )

switch ( a < 4 && b > 7 )
```

Avoid

•A float expression cannot be tested using a **switch**
•Cases can never have variable expressions (for example it is wrong to say **case a +3 :)**
•Multiple cases cannot use same expressions. Thus the following **switch is illegal:**

```
{
case 3 :
...
case 1 + 2 :
...
}
```

Using switch- case

a) WAP that will output following

   x= a+b if y=2
   x= a-b if y=4
   x= a*b if y=0
   x= a/b if y= 1
   x= a%b, otherwise

```
#include<stdio.h>
void main()
{
int y,a=5,x,b=2;
scanf("%d",&y);
switch(y)
{
case 2: x=a+b;break;
case 4: x=a-b;break;
case 0: x= a*b;break;
case 1: x=a/b;break;
default: x=a%b;
}
printf("x=%d",x);
}
```

Using switch- case

a) WAP that will output following

    x= a+b if y=‘+’

    x= a-b if y=‘-’

    x= a*b if y=‘*’

    x= a/b if y= ‘/’

    x= a%b, otherwise

```
#include<stdio.h>
void main()
{
int a=5,x,b=2;
char y;
scanf("%c",&y);
switch(y)
{
case '+': x=a+b;break;
case '-': x=a-b;break;
case '*': x= a*b;break;
case '/': x=a/b;break;
default: x=a%b;
}
printf("x=%d",x);
}
```

Problem

WAP that will calculate area of the following entities.

- If user enter value 1,your program will display area of circle
- If user enter value 2,your program will display area of triangle
- If user enter value 3,your program will display area of rectangle
- Other wise your program will display “Please enter either 1 or 2 or 3”

Problem

WAP that will display days of a week based upon the value entered by user

For example : if value is 1,it will display SUNDAY

if value is 2,it will display Monday and so on,

Solve same problem using if-else-if ladder.

```
#include<stdio.h>
#include<conio.h>
#define PI 3.1416
main()
{
int choice,length,breadth,base,height;
float radius,area;
printf("Enter value 1 or 2 or 3
to\ncalulate either area of circle\n or
area of triangle or \narea of
rectangle");
scanf("%d",&choice);
switch(choice)
{
case 1: printf("Enter radius");
            scanf("%f",&radius);
            area=PI*radius*radius;
            printf("area of circle
is=%f",area);
            break;
case 2: printf("Enter base and height");
            scanf("%d %d",&base,&height);
            area=1.0/2*base*height;
            printf("area of triangle
is=%f",area);
            break;
case 3: printf("Enter length and
breadth\n");
            scanf("%d
%d",&length,&breadth);
            area=length*breadth;
            printf("area of rectangle
is=%f",area);
            break;
default: printf("donot press any key except 1
or 2 or 3");
}
return 0;
}
```

```
#include<stdio.h>
#include<conio.h>
#define PI 3.1416
main()
{
int choice,length,breadth,base,height;
float radius,area;
printf("Enter value 1 or 2 or 3
to\ncalulate either area of circle\n or
area of triangle or \narea of rectangle");
scanf("%d",&choice);
        if(choice==1)
        {
        printf("Enter radius");
        scanf("%f",&radius);
        area=PI*radius*radius;
        printf("area of circle
is=%f",area);
        }
else if(choice==2)
        {
        printf("Enter base and height");
        scanf("%d %d",&base,&height);
        area=1.0/2*base*height;
        printf("area of triangle is=%f",area);
        }
else if(choice==3)
        {
        printf("Enter length and breadth\n");
        scanf("%d %d",&length,&breadth);
        area=length*breadth;
        printf("area of rectangle
        is=%f",area);
        }
else
printf("donot press any key except 1 or 2 or 3");
return 0;
}
```

Common Programming error

```
main( )
printf("Hello World");
}
```

```
main( )
{
printf("Hello World")
}
```

```
main( )
{

printf("Enter a value");
scanf("%d",a);
Printf("Value is %d");
}
```

```
main( )
{
int a;
printf("Enter a value");
scanf("%d",&a);
printf("Value is %f",a);
}
```

Common Programming error

```
scanf(“%d %d %d”,a,b,c);
printf(“%d %d %d %d”,a,b,c,d)
```

```
main( )
{
int a;
float b;
scanf(“%d %d”,&a,&b);
if(a>b);
printf(“%d”,a);
else
printf(“%d”,b);
}
```

```
main( )
{
int a,b
scanf(“%d %d”,&a,&b);
if(a>b)
printf(“%d”,a);
printf(“I am a”);
else
printf(“%d”,b);
printf(“I am b”);
}
```

```
main( )
{
int a;
float b;
scanf(“%d %d”,&a,&b);
printf(“%d %d”,a,b);
}
```

Contd...

What are outputs of the following code?

```
main()
{
int a=10,b=20;
if(a>b);
{
a=a+15;
b=b+25;
}
printf("%d %d",a,b);
}
```

What are outputs of the following code?

```
main()
{
int a=10,b=20;
if(0)
{
a=a+15;
b=b+25;
}
printf("%d %d",a,b);
}
```

What are outputs of the following code?

```
main()
{
int a=10,b=20;
If(255)
{
a=a+15;
b=b+25;
}
printf("%d %d",a,b);
}
```

If(3) {….} here 3 is non zero value so whatever code is written inside curly brackets will be execute.

If(0) {…} 0 means condition false the code written inside curly brackets will not be execute.

Thank You