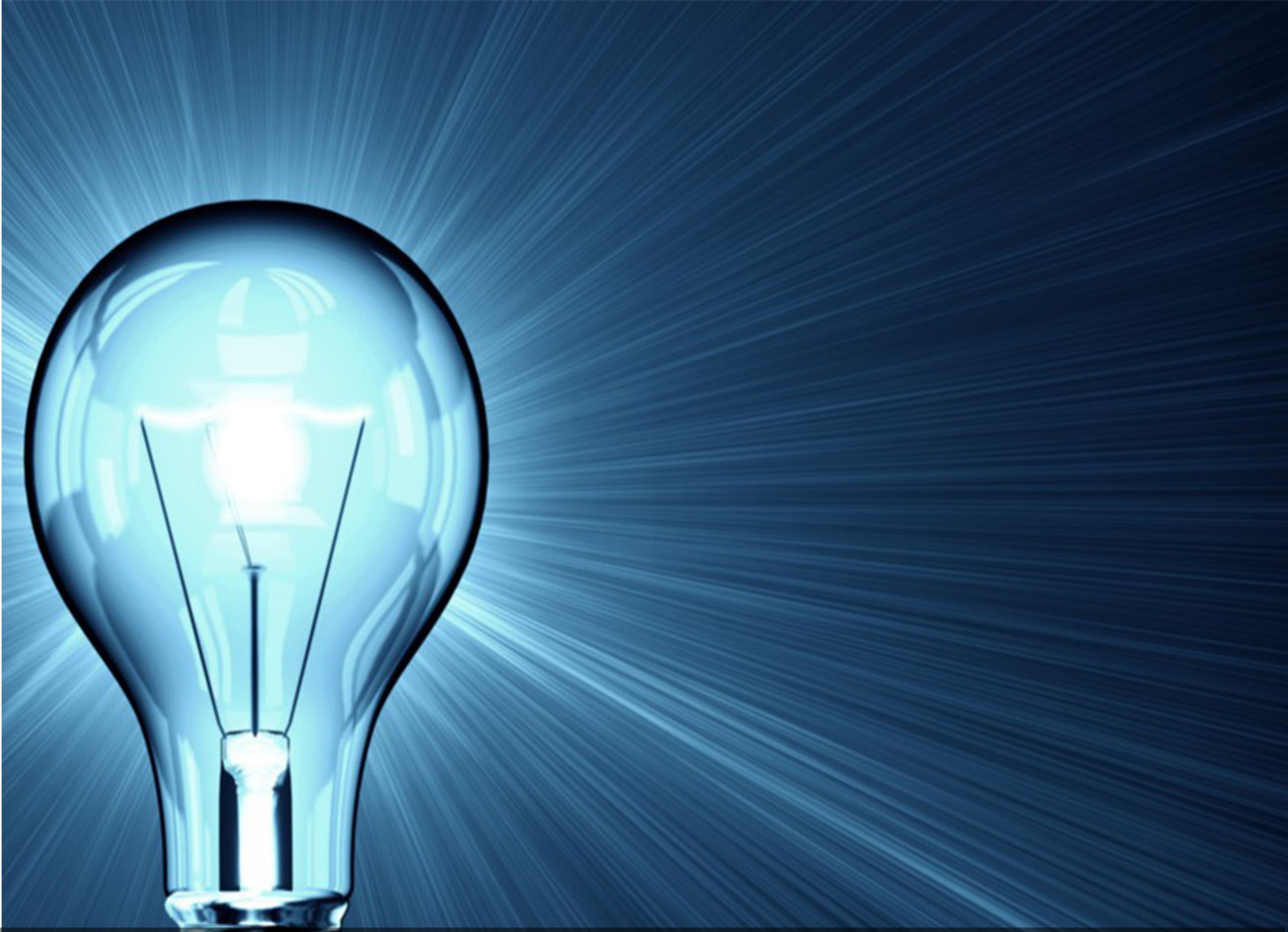


STRUCTURED PROGRAMMING APPROACH

SEM - II - All Branches

As per revised syllabus of Mumbai University

Prof. Sameer Velankar

GLORIOUS ACADEMY

Success is a process not an event, so come be part of process

DEGREE & DIPLOMA ENGINEERING CLASSES

www.gloriousacademy.org

Contents

Structure of simple C program , operators and expressions

Structure of simple C program

A simple C program is given below
program

```
1. #include <stdio.h>
2. void  main()
3. {
4.  printf( "hello everyone\n");
5. }
```

The above program is a set of 5 lines . This program will display output as shown in fig.

```
hello everyone
```

Every C program starts with statement
#include <stdio.h>
and this line is followed by the statement
void main()

The meaning of these two lines is difficult to understand in chapter 1. So we delay the study of these 2 lines till later chapters. So reader is suggested to blindly start every C program with the above mentioned lines without knowing the importance of the two lines at this moment. Line 3 is the open brace bracket and line 5 is the close brace bracket. The statements are written in between open and close brackets. Each statement ends with semicolon(;). The line 4 of the program is used for displaying the output

hello everyone

on the screen.
The symbol \n given at the end of "hello everyone\n" is used for taking cursor to the next line. \n is a special character which will get cursor on the next line.
Before we go ahead let us look at printf statement again. Consider the following printf statement
printf("3 + 4");
The output due to above printf will be 3 + 4 and not 7.
Well, that means whatever is given in the pair of quotation mark in printf statement is printed as it is.
Let's take some more examples of printf statement.

Program

```
1. #include <stdio.h>
2. void main()
3. {
4.  printf( "hello\n");
5.  printf( "world");
6.  printf("welcome");
7. }
```

Output of the program is given in fig

```
hello
worldwelcome
```

First printf prints word hello and \n takes the cursor to the next line. the second printf prints the word world and keeps the cursor on the same line. The third printf prints word welcome on the same line as world.

What is output of following printf statements.

(a)	(b)

printf("wel\ncome");	printf("welcome");
	printf("\nto");
	printf("\nC");

Answer wel come	Ans welcome to C

Now that we have seen basic structure of C program , our next step will be to see some more important terms such as *variable* and *datatypes.*

Variable

It is a quantity that can change during the execution of program. Basically, variable is not a new concept for anyone. some basic examples of variables in day to day life are
(a) Our **age** is a variable as it is changing at every moment.
(b) My **salary** is a variable. I wish that it should always increase (By large amounts)

In programming we need a variable to store a value. For example in a computer game, **score** is a variable that is initially zero. As we play the game, the score either increases or decreases.
Variable name : Every variable has a name and there are some rule regarding naming a variable.
rule (a) : name of variable must start with alphabet or underscore.
rule (b) name of variable can contain only alphabets, digits and underscores.

Example of valid variable names are

1) a 2) a1 3) first 4) roll_no 5) roll_no_of_student

Examples of invalid variable names are

1) 1a (name cannot begin with a digit)
2) abc,def (name cannot contain comma)
3) abc def (name cannot contain space)

Data type

Data type stands for type of data. We all know that computer can deal with numerical data and character data (like name of a person)

C supports four basic datatypes which are used to specify type of data that a variable can store.

(a) int : This data type is used for storing integers(or whole numbers) such as 15, -3 , 0, 35 and so on. Consider the following statement
int x;

The above line is called declaration of a variable. Here variable x is declared of type int. So variable x can now store only integer type of value. Note that real numbers like 3.5 or -23.8 cannot be stored by int variable. Range of int type is -32768 to 32767.

Variable of int type consumes 2 bytes of memory.

(b) float : This data type is used for storing real numbers(or floating point values) such as 15.8, -3.67 , 0, 35.2 and so on.

Consider the following statement

float y;

The above line is called declaration of a variable. Here variable y is declared of type float. So variable y can now store any real number. Note that real number like 3 can also be stored by float variable. In such case 3 will be stored as 3.0 automatically.

Range of float data type is $3.4 * 10^{-38}$ to $3.4 * 10^{38}$. This means that a float variable can store value with the above specified limits.

Variable of float type consumes 4 bytes of memory.

(c) double : This data type is also used for storing real numbers. So double is actually same as float. but a variable of double data type can store larger real number than that a float variable. In other words range of double data type is larger than float data type.

Range of double data type is $1.7 * 10^{-308}$ to $1.7 * 10^{308}$ (That is a huge range of values)

This is why double data type is also called long float.

Variable of double type consumes 8 bytes of memory.

(d) char : This data type is used for storing one character such as 'a' or 'Z' etc.

The line

char t;

declares a variable t of type char. This variable t can store a single character.

Variable of char type consumes 1 bytes of memory.

Comments or remarks in a C program

Comments can be added in program by starting any line with // symbol. Any line which begins with // symbol is marked as comment and such lines are not executed. We usually write comments to explain our program to other readers or such comments act as a good documentation or description of the program.

There are two ways of giving comments in C

Method 1 : Single line comment.

Start the line which you want to be comment with // symbol.

Example

// this is a single line comment this will not execute

Method 2 : Multiline comment.

Enclose the multiple lines which you want to be comment in the pair of /* and */ symbol.

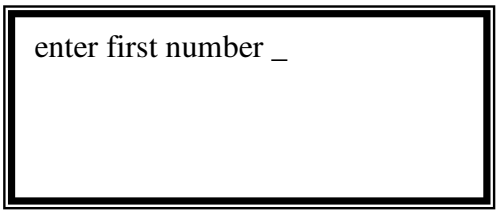
Example :

/* this is a multiline comment
this will not execute */

Program
The program given below will accept two integers from keyboard and find their sum.

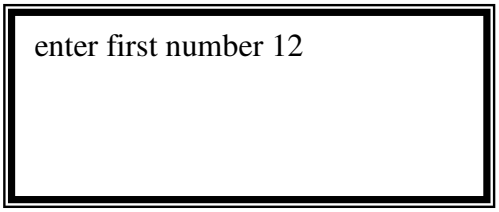
```
1.    #include <stdio.h>
2.    void main()
3.    {
4.          int  a,b,c;
5.          printf("enter first number ");
6.          scanf("%d" , &a);
7.          printf("enter second number ");
8.          scanf("%d" , &b);
9.          c = a + b;
10.         printf("%d" , c) ;
11. }
```

Lines 1 and 2 are always the same in all programs and meaning of these lines will be explained in the later chapters.
Line 4 is the declaration of three variables a,b and c. All three variables are declared of type int.
Line 5 will print the specified message on the screen.

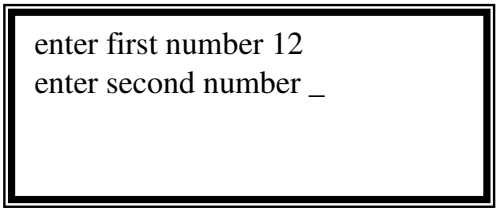


The cursor will remain on the same line after the message as \n was not given in the printf statement.

Line 6 will allow us to input a number through keyboard. This number entered will automatically get stored in variable 'a'. scanf statement allows user to input a value through keyboard.



Line 7 will again print the specified message on the screen.



The cursor will remain on the same line after the message as \n was not given in the printf statement.

Line 8 will allow us to input a number through keyboard. This number entered will automatically get stored in variable 'b'.

```
enter first number 12
enter second number 13
```

line 9 is called assignment statement as c = a+b is read as "c is assigned a value of a+b"
line 10 will finally output value of variable c

```
enter first number 12
enter second number 13
25
_
```

Let's have a close look at the last printf statement. Check that the printf statement is not given as
printf("c") ;
Well this will print the letter c on the screen. We are not interested in printing letter c but we want the value of variable c. hence the printf is given as
printf("%d" , c);
This printf statement uses the format "%d" to print value of integer variable c.
Suppose we needed the output of the above program as
sum = 25
Then the printf statement will be
printf("sum = %d" , c);

and suppose we needed the output as
sum of 12 and 13 = 25
Then the printf statement will be
printf("sum of %d and %d = %d" , a ,b , c) ;

program

```
1 /*program to input radius and output area of circle */
2 #include <stdio.h>
3 void main()
4 {
5   float r,a;
6   printf( "enter radius ");
7   scanf("%f" , &r);
8   a = 3.14 * r * r;
9   printf("area = %f " , a) ;
10 }
```

Line 1 is a comment which describes what the program does.

Lines 2 and 3 are always the same in all programs and meaning of these lines will be explained in the later chapters.
Line 5 is the declaration of two variables r and a . Both the variables are declared of type float. This is because radius of circle can be a real number like 2.5 and hence r is declared as
float r;

The another variable 'a' is declared for storing area of the circle. The variable 'a' must be float as area will be calculated as 3.14 * r * r.

Line 6 will print the specified message on the screen.

```
enter radius _
```

The cursor will remain on the same line after the message as \n was not given in the printf statement.

Line 7 will allow us to input a value through keyboard. This value entered will automatically get stored in variable 'r'. scanf statement allows user to input a value through keyboard.

```
enter radius 2.5
```

Line 8 is called Assignment statement. This statement assigns value of 3.14 * r * r to the variable a. Hence variable a now stores area of circle.

Line 9 finally prints the output

```
enter radius 2.5
area = 19.625
```

program

```
/*program to input length and breadth and output area of rectangle*/
#include <stdio.h>
void main()
{
  float l, b, a;
  printf( "enter length ") ;
  scanf("%f" , &l);
  printf( "enter breadth ") ;
  scanf("%f" , &b);
  a = l * b;
  printf( "area = %f " , a) ;
```

```
}
```

```
enter length 2.2
enter breadth 3.2
area = 8.75
_
```

Exercise
Q . program to find area of triangle using base and height

1.5 Mathematical or Arithematical operators in C

There are five Mathematical or Arithematical operators in C
+ addition operator
- subtraction operator
* multiplication operator
/ division operator
% remainder operator

Out of the above mentioned operators only division(/) and remainder (%) operator need explaination.

Division operator :
Very important rule about division operator is as follows
If integer value is divided with other integer value , the result is always an integer.
But if either of the two values which are getting divided are floats then result result is a float.

Example

5 / 2 is 2 and not 2.5

1028 / 10 is 102 and not 102.8

2 / 5 is 0 and not 0.4

This shows that if numerator and denominator are integer values , then the result of division is always integer.

5.0 / 2 is 2.5
Note numerator is a float so result of division is a float that is 2.5

1028 / 10.0 is 102.8
Note denominator is a float so result of division is a float that is 102.8

2.0 / 5.0 is 0.4
Note both numerator and denominator are float so result of division is a float that is 0.4

This shows that when floating point numbers are divided , then the result of division is always a float.

Remainder operator (%)

This operator gives remainder after dividing two integers.

5 % 2 is 1.

1028 % 10 is 8

2 % 5 is 2

There are some things worth noting about % operator
i) Any integer n % 10 will always give last digit of that integer as result. Example 127 % 10 = 7

ii) If LHS of % is less than RHS then result is always equal to LHS. Example 11 % 12 = 11

Increment(++) and Decrement (--) operator

Two additional operators are given by C to increment and decrement a value by 1.

Example of ++ operator

```
int a = 1;
a++;
printf("a = %d " , a) ;
```

Output of the above code segment is
2

The statement

```
a++ ;
```

in the above code segment is same as

```
a = a + 1;
```

so a++ really means that value of variable a should be increased by 1.

Example of -- operator

```
int a = 1;
a--;
printf( "a = %d " , a ) ;
```

Output of above code segment is
0

The statement

a-- ;

in the above code segment is same as

a = a - 1;

so a-- really means that value of variable a should be decreased by 1.

Note : ++ and – are just shortcut operators given so that we can avoid typing longer expressions. Consider an expression again

result = result + 1;

The above expression can be equivalently typed as

result++;

So we have reduced our work of typing the word result twice..

Like ++ and -- operators , many other shortcut operators are provided by C.
Here are some more such shortcut operators.

Shortcut Operator	Example	Meaning
+=	a += 3;	a = a + 3;
-=	a -= 3;	a = a - 3;
*=	a *= 3;	a = a * 3;
/=	a /= 3;	a = a / 3;
%=	a %= 3;	a = a % 3;

Now that we have seen Mathematical operators , let us see priority of these five mathematical operators.

Out of 5 mathematical operators (* / % + -) the highest priority is given to * / and % operators. In fact * / and % have same priority. So whichever of these operators come first in an expression is solved first.
+ and – operators also have same priority but + and – have lesser priority than * / and %.
Let us see some examples of mathematical expressions.

Example 1

int a = 5, b = 14 , c = 7 , d;
d = a + b / c;

Here, first b/c will be solved as division has higher priority than addition. The expression will therefore get solved as

```
d = 5  + 14 / 7;
d = 5  +  2;          Note 14/7 gets solved first
d = 7;
```

Hence final value of d is 7

Example 2 :

```
int a = 5, b = 14 , c = 7 , d = 8 , e;
e = a + b / c - d;
```

Here , first b / c will be solved as division has higher priority than addition and subtraction. The expression will therefore get solved as

```
e = 5 + 14 / 7 – 8;
e = 5 + 2 – 8;          Note 14/7 gets solved first
e = 7 – 8;
e = -1;
```

Hence final value of e is -1.

Exercises

Q 1 Rewrite the following expressions in C

a) $\frac{a + b}{c+d}$	b) $\frac{a^2-2a}{c} * \frac{d}{e}$	c) $\frac{(a-b)^2}{(c-d)^2}$

Q 2 State the output of following

a)
```
int a = 10 , b = 5 , c = 20 , d = 8 , e;
  e = c + a * d % b;
  printf( "e = %d" , e);
```

b)
```
int a = 10 , b = 5 , c = 20 , d = 8 , e;
  e = a / d + b % c;
  printf(  "e = %d" , e);
```

c)
```
int a = 7 , b = 15 , c = 2 , d = 88 , e , f;
  e = a + b / c - d;
  f = a  + b  % c / d;
  printf( "e = %d  f = %d"  , e ,  f);
```

d)
```
int a = 17 , b = 10 , c = 25 , d = 8 , e , f;
  e = a / b / c - d;
  f = a  *  b  % c / d;
  printf( "e =  %d  f = %d" ,  e , f);
```

Q 3 Suppose that we have an expression

$$\frac{a^2 + b^2}{c^2 + d^2}$$

then which of the following C expressions are valid for the above expression.

a) z = a* a + b * b / c*c + d*d;
b) z = (a* a) + (b * b) / (c*c) + (d*d);
c) z = ((a* a) + (b * b)) / ((c*c) + (d*d));

Q 4 Write a program to input coordinates of end points of a line and output slope
Q 5 Write a program to input three sides of triangle and output area
Q 6 Write a program to input coordinates of end points of a line and output distance
Q 7 Write a program to find area of triangle using base and height

Additional Exercises

Q1 Write a program to input temperature in celcius and output temperature in farenheit.
Q2 Write a program to input voltage and resistance and output value of current.
Q3 Write a program to input principal amount (p), rate of interest (r) and number of years (n) and output amount after n years.
Q4 Write a program to input major and minor axis of ellipse and output area of ellipse.
Q5 State the errors in following programs

a)
```
1 #include <stdio.h>
2 void main()
3 {
4  int  a = 10 , b = 20 ;
5  c = a + b;
6  printf( "c = %d" , c);
7 }
```

b)
```
1 #include <stdio.h>
2 void main()
3 {
4   int  a =10 ,b = 20 ,c = 5 ;
5   int  d = 2 , e;
6     e = (a * ( b / c ) – d ;
7     printf("e = %d" , e) ;
8 }
```

c)
```
1 #include <stdio.h>
2 void main() ;
3    int  a = 10 ;
4   printf( "a = %d" , a) ;
5 }
```

d)
```
1 #include <stdio.h>
2 void main()
3 {
4    int  a = 10 ;
5    printf( "a = %d"  a) ;
6 }
```

e)
```
1 #include <stdio.h>
2 void MAIN()
3 {
4    int  a = 10;
5    PRINTF( "a = %d"  ,  a) ;
6 }
```

Q 6 Which of the following are valid and invalid variable names.
a) first b) roll_no c) 12_abc
d) ab-cd e) department_1

Q 7 What should be printf statement in the following program if the output needed is
4 * 7 = 28

```
#include <stdio.h>
void main()
{
   int  a = 4 , b = 7 , c;
   c = a * b;
```

```
   printf( _____);
}
```

Q 8 State the output of following

```
#include <stdio.h>
void main()
{
  int a = 4 , b = 7 ;
  printf( "a = %d\n"  ,  a);
  printf( "b = %d\n"  ,  b);
  printf("%d  *  %d = %d\n" , a ,b ,c) ;
}
```

Q 9 Give the order in which operations will be carried out in the following arithematical expressions.
a) t + x * y / z ;
b) a + b / c + d * e
c) a * a – b * b / c * c – d * d

CONTROL STRUCTURES
DECISION MAKING

The programs that we saw in chapter 1 were very basic and primary program. In such programs the control typically goes from first line to last line. That is the program is executed in line after line in sequential order.
How ever in many other programs we would like to execute certain statements depending on some condition. In other words it is required to shift the control of the program abruptly from a particular line to some other line. In many situations we would not like the program to be executed sequentially but we would like to select or skip certain statements. This work is accomplished by control structures.

As an example let us consider the program for finding slope of line discussed earlier in chapter 1

program

```
/*program to input coordinates of end points of a line and output slope */
1 #include <stdio.h>
2 void main()
3 {
4   float x1, y1, x2, y2, m;
5   printf("enter co ordinates of first point ") ;
6   scanf("%f %f" , &x1 , &y1);
7   printf("enter co ordinates of second point ") ;
8   scanf("%f %f" , &x2 , &y2);
9   m = (y2-y1)/(x2-x1);
10  printf("slope = %f " , m );
11 }
```

If we take a close look at the above program , we would certainly find a possible error. At first look reader may get surprised to know that above program may result in error. This error situation will arise if x2 = x1. In such case the line 9 will be calculated as

m = (y2 – y1) / 0; AS x1 = x2

As everyone knows that division by zero is undefined. So what do we do about this error. The answer is to use if else control structure.

The fig shows what happens if value of x1 is equal to x2.

```
enter coordinates of first point  2 2
enter coordinates of second point 2 3
floating point error: Divide by 0
Abnormal program termination
```

if else control structure

This control structure is used for selecting or skipping certain statement depending on certain condition.

General syntax of if else statement is

```
if (condition)
  statement1;
else
  statement2;
```

Here condition is any boolean expression which will be evaluated to either true or false(For example x > y or salary > 10000). statement1 and statement2 are any valid C statement. The if else statement shown above means that statement1 should be executed if condition written in the brackets is true and statement2 should not be executed. On the other hand statement2 should be executed if condition is false.

The above program can be rewritten using if else as follows

program

```
/*program to input coordinates of end points of a line and output slope */
1 #include <stdio.h>
2 void main()
3 {
4   float x1, y1, x2, y2, m;
5   printf("enter co ordinates of first point ") ;
6   scanf("%f %f" , &x1 , &y1);
7   printf("enter co ordinates of second point ") ;
```

```
8  scanf("%f %f" , &x2 , &y2);
9  if ( x1  == x2 )
10 printf("Line is perpendicular to y axis  slope is infinity ");
11 else
12 {
13  m = (y2-y1)/(x2-x1);
14  printf("slope = %f " , m) ;
15 }
16 }
```

The program starts by accepting values of x1,y1 (coordinates of first point of the line) and x2,y2 (coordinates of second point of the line). Once this is done the program checks if x1 is equal to x2. If this is true then the message "Line is perpendicular to y axis so slope is infinity " is displayed on the screen.

But if x1 is not equal to x2 then the program finds slope m and also displays it (Line 13 and 14). The fig shows what happens when the above program is executed. Note that this execution is much better than the execution of program 2.1 which simply shows an error "floating point error divide by 0".

```
enter coordinates of first point  2 2
enter coordinates of second point 2 3
line is perpendicular to x axis so slope is infinity
```

Now we will look into more details of if else statement and will explain various ways to use if else statement.

Let us consider a program of checking whether a number is even or odd. The program being solved here is very simple. We all know that 12 is even because it is perfectly divisible by 2. Where as 13 is an odd number because 13 is not divisible by 2.

```
// program to check if number is even or odd
#include <stdio.h>
void main()
{
 int  x;   // x stores a number
 // input the number
 printf("enter a number ");
scanf("%d" , &x);

 if (x % 2 == 0)
  printf("even number ");
 else
  printf("odd number ");
}
```

Now let us consider a program which accepts 2 numbers and print largest among the two numbers

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

```
void main()
{
  int x, y;  // x and y will store two numbers
  // input the two numbers
   printf(" enter two numbers ") ;
   scanf("%d %d" , &x , &y);
   // check is x greater than y
   if ( x  >  y)
     printf("%d" , x) ;   // output the value of x if x > y is true
   else
     printf("%d" ,  y);  // output the value of y if x > y is false

 }
```

Logical Operators

The previous section discussed some fundamental program which use simple if else statement which decide whether to execute a statement or not. But we don't get such simple situations or conditions always. Often we have to check a complicated condition. For example we would like to check if salary of an employee is between Ten thousand and Twenty thousand. In such complicated condition we have to use LOGICAL OPERATORS.

There are three logical operators in C.
1) && which is read as AND
2) || which is read as OR
3) ! which is read as NOT

These three operators can be best understood with some simple examples

&& operator : Let us say that we want to check if marks obtained by a student is between 60 and 75 (both inclusive). Such a condition should be written as

```
if ( marks >= 60  && marks <= 75)
```

The above condition will be true only if both the conditions marks >= 60 and marks <= 75 are true. In other words both the conditions must be true to make the entire condition true. We can check ourselves what happens if marks is equal to 70. as marks is 70 the condition becomes

```
if ( 70 >= 60 && 70 <= 75)
```

Here both the conditions are true. Hence the overall condition in if statement will be treated as true. Again consider what happens if marks is equal to 90. In such case the condition becomes

```
if (90 >= 60 && 90 <= 75)
```

Here the first condition 90 >= 60 is true whereas the second condition 90 <= 75 is false. Hence the overall condition in if statement will be treated as false.

You may have already understood use of && operator. But we can further understand && operator by looking at the following definition.

&& is a logical operator which is used to join two boolean expressions (conditions) in such a way that the entire condition is treated true only if both the conditions are true.
Let C1 and C2 be the two conditions joined using && operator. Then following table shows the result of ANDing both C1 and C2 conditions.

Condition C1	Condition C2	Result of C1 && C2
False	False	False
True	False	False
False	True	False
True	True	True

|| operator : Let us say that we want to check if grade of an employee is 1 or 2. Such a condition should be written as

if (grade = = 1 || grade = = 2)

The above condition will be true if at least one of the conditions grade = = 1 or grade = = 2 are true. In other words at least one condition joined using | | operator should be to make the entire condition true. We can check ourselves what happens if grade of an employee is 2 . Putting the actual value of grade in the condition we have

if (2 = = 1 || 2 = = 2)
Here first condition is false but second condition is true. Hence the overall condition in if statement will be treated as true. Again consider what happens if grade of an employee is 3. Putting the actual value of grade in the condition we have

if (3 = = 1 || 3 = = 2)
Here both the conditions are false. Hence the overall condition in if statement will be treated as false.

You may have already understood use of || operator. But we can further understand || operator by looking at the following definition.

|| is a logical operator which is used to join two boolean expressions (conditions) in such a way that the entire condition is treated true only if at least one condition is true.
Let C1 and C2 be the two conditions joined using || operator. Then following table shows the result of ORing both C1 and C2 conditions.

Condition C1	Condition C2	Result of C1 || C2
False	False	False
True	False	True
False	True	True
True	True	True

! operator (NOT operator)

This operator is used for negating a condition. Not operator converts a true condition into false and false condition into true.
Some people have a habit of stating a condition using NOT. as a simple example consider a condition in plain English language

if SALARY is greater than or equal to 10000 ……. 1

The above condition can be equivalently stated as

if SALARY is NOT below 10000 ……. 2

Both the conditions are actually same but are stated in different manner. Statement 2 can be written in C as

```
if ( ! (salary < 10000) )
```

which is same as

```
if ( salary >= 10000)
```

To further understand ! operator consider that salary of an employee is 20000. Putting this value of salary in statement 3 we get

```
if ( ! ( 20000 < 10000))
```

Check that the condition 20000 < 10000 is false. but NOT of false is true. hence the condition in the if statement will be true. On the other hand consider that salary of an employee is 8000. Putting this value of salary in statement 3 we get

```
if ( ! (8000 < 10000))
```

Check that the condition 8000 < 10000 is true. but NOT of true is false. Hence the condition in the if statement will be false.

Exercises

1. Write the following conditions in C language

a) check whether a is greater than b AND c.

Ans

```
if ( a > b  &&  a > c)
```

b) check whether x is zero OR one OR two

Ans

```
if ( x = = 0  ||  x = = 1 || x = = 2)
```

c) check if z is less than zero OR more than five

Ans

```
if ( z < 0 || z > 5)
```

2. Write a program to input age and weight of a person and check whether the person is eligible for blood donation or not. The person is eligible only if age is between 18 and 65 and also weight is at least 50.

Ans

```
#include <stdio.h>
void main()
{
  int age , weight;  //declare variables for storing age and weight

  //input age and weight
  printf("enter age of person ");
  scanf("%d" , &age);

  printf("enter weight in kgs ");
  scanf("%d" , &weight);
  //now check for eligibility
  if ( age >= 18 && age <= 65  && weight >= 50)
    printf("person is eligible");
  else
    printf("person is not eligible");
}
```

3. Assume that a = 10 , b = 25 and c = 35. Then state whether following boolean expression will be true or false

a) a > b && a > c
b) a > b || a > c
c) a < b || a > c
d) a + b = = c || a > b + c
e) a = = b || c > b
Ans
a) false b) false c) true d) true e) true

Nested if else statement

Nested if else statement is nothing but if statement nested inside another if statement. We use nested if else statement many times in day to day life. As an example consider following scenario

```
if  Annual salary  is greater than 5 lacs then
   income tax should be 30 % of annual salary
    else
       if  Annual salary  is less than 5 lacs but greater than 3 lacs then
                          income tax should be 10 % of salary
       else
          if  Annual salary is less than 3 lacs but greater than 1 lacs
                          income tax should be  5 % of salary
          else
                          income tax should be zero
```

The above nested if else statement is simple and it is a statement which we all use in day to day life. Nested if else is also some times called ***ladder if*** statement. Checking the above nested if else we understand that income tax will be zero if all the if statements are false(that is if annual salary is less than or equal to 1 lacs)

Syntax or general grammar of nested if else statement

```
if (condition1)
  statement1 ;
else
  if (condition2)
   statement2;
else
  if (condition3)
   statement3;
 else
   statement4;
```

The above grammar can be read as
if condition1 is true then execute statement1.
But if condition1 is false only then check the condition2. if condition2 is true then execute statement2.
And so on it continues.
Note that statement4 will execute only if all the three conditions are false.

Now we see many programs which use nested if else statement.

Exercises

1. Write a program to input three integers a , b and c and print largest among them.

```
#include <stdio.h>
void main()
{
    int a,b,c;  //variables a , b and c will store three integers
          // input three integers
          printf("enter three integers ");
          scanf("%d %d %d" , &a , &b , &c);
          // now check the largest
          if (a >= b && a >= c)
                    printf("%d" , a);
          else
                    if ( b >= a && b >= c)
                              printf("%d" , b);
                    else
                              printf("%d" , c);
          }
```

2. Write a program to input coordinates of a point x,y and then check in which quadrant the point lies.

3. Write a program to input center coordinates of a circle x,y and radius of circle r. Also input coordinates of a point x1,y1 and then check whether the point lies inside , outside or on the circle.

4. Write a program to input an integer and check if the number is even or odd. Also check if the integer is negative or zero. This is because negative numbers are neither even nor odd.

5. Write a program to input marks scored by a student in three subjects and calculate total of three subject marks. Then display result using following criteria

if marks of any subject is less than 40 then result is FAIL
else
if total is 75 or more then result is distinction
else
if total is 60 or more then result is first class
else
result is second class

6. Write a program to input price of an article and discount policy. Discount is offered using following criteria.
If policy if 1 then discount = 10 % of price
If policy if 2 then discount = 20 % of price
If policy if 3 then discount = 30 % of price
Calculate the actual price.

Solution :

```
#include <stdio.h>
void main()
{
 int policy;
 float price , discount , finalamt;

 printf("Enter price ");
 scanf("%f" , &price);

 printf("Enter policy ");
 scanf("%d" , &policy);

 if (policy == 1)
        discount = 0.1 * price ;
 else
        if (policy == 2)
          discount  = 0.2 * price;
        else
          if (policy == 3)
           discount = 0.3 * price;
          else
           discount = 0;
```

```
  finalamt = price - discount;
  printf("Final amount = %f" , finalamt);
}
```

7. Write a program to input 3 sides of a triangle and find whether the 3 sides form a triangle or not. If the 3 sides form a triangle then also find whether it is right angle triangle, or isosceles triangle, or equilateral triangle or scalian triangle.

Switch case control structure

Switch case is a control structure which is same as if else or nested if else. In fact switch case is a neater method to write a nested if else statement.
Syntax of switch case is

```
switch (expression)
{
 case value : statement;
                  break;
 case value : statement;
                   break;
 case value : statement;
                   break;
  “       “       “
  “       “       “
 default : statement ;
 }
```

Q 1: Write a program to input price of an article and discount policy. Discount is offered using following criteria.
If policy if 1 then discount = 10 % of price
If policy if 2 then discount = 20 % of price
If policy if 3 then discount = 30 % of price
Calculate the actual price.
Solution :

```
#include <stdio.h>
void main()
{
 int policy;
 float price , discount , finalamt;

 printf("Enter price ");
 scanf(“%f” , &price);

 printf("Enter policy ");
 scanf(“%d” &policy);

 switch(policy)
```

```
 {
  case 1 : discount = 0.1 * price ;
                        break;
  case 2 : discount  = 0.2 * price;
                        break;
  case 3 : discount  = 0.3 * price;
                        break;
  default : discount = 0;
 }

  finalamt = price - discount;
  printf("Final amount = %f" , finalamt);
}
```

Q 2: WAP to input month number and print number of days in that month.

Solution :

```
#include <stdio.h>
void main()
{
 int month , year;

 printf("Enter month number ");
 scanf("%d" , &month);

 switch(month)
 {
          case 1 :
          case 3 :
          case 5 :
          case 7 :
          case 8 :
          case 10 :
          case 11 : printf("days = 31 ");
          break;

  case 4:
  case 6:
  case 9:
  case 11: printf("days = 30 ");
        break;

  case 2 : printf("enter year ");
        scanf("%d" , &year);
        if (year % 4 == 0)
          printf("days = 29 ");
        else
          printf("days = 28 ");
        break;
```

```
  default : printf("Invalid month number ") ;
 } // end of switch
} // end of main
```

Q 3 : A company manufactures 3 goods and have given category for each good. The discount offered by the company on various goods is as per following criteria.

Category	CONDITION	Discount offered
1 (Television)	Price is between 50000 and 100000 Price is between 20000 and 49999 For price lower than 20000	8 % of the price 2 % of the price No discount
2 (Music system)	Price is between 20000 and 40000 For price lower than 20000	2 % of price Discount = 0.5 % of price
3 (refrigerator)	For any price	Discount = 0

Wap to input the category of the product sold and price of that product. Then using the above rules, find out the discounted price.(Hint use both if else and switch case)

Q 4: A company has categorized its employees in 4 major categories. The salary is calculated using following rules

Category	Salary
1	Rs 100 for every regular hour and no overtime
2 or 3	Rs 150 for every regular hour and 1.5 times the rate of regular hour for every overtime hour.
4	Rs 200 for every regular hour and 2.5 times the rate of regular hour for every overtime hour.

WAP to input category of the employee, regular hours and overtime hours. Then calculate salary using the above rules.

Q 5 : State the output of following :

```
#include <stdio.h>
void main()
{
int x = 1;

switch(x)
{
case 1 : printf("one");
case 2 : printf("two");
case 3 : printf("thr");
default: printf("def");
}

}
```

```
#include <stdio.h>
void main()
{
int x = 2;

switch(x)
{
case 1 : printf("one");
case 2 : printf("two");
case 3 : printf("thr");
default: printf("def");
}

}
```

```
#include <stdio.h>
void main()
{
int x = 2;

switch(x)
{
case 1 : printf("one");
case 2 : printf("two");
break;
case 3 : printf("thr");
default: printf("def");
}

}
```

Q 6 : Convert following nested if else statements into switch case

```
#include <stdio.h>
void main()
{
int x = 11;

if ( x == 10)
  statement 1 ;
else
if ( x == 12)
  statement 2 ;
else
  statement 3;
}
```

```
#include <stdio.h>
void main()
{
int x = 11;

if ( x == 10)
{
  statement 1 ;
  statement 2;
  statement 3;
}
else
if ( x == 12)
{
  statement 4 ;
  statement 5 ;
}
else
Statement 6 ;
}
```

```
#include <stdio.h>
void main()
{
int x = 11;

if ( x > 10)
{
  statement 1 ;
  statement 2;
  statement 3;
}
else
if ( x == 12)
{
  statement 4 ;
  statement 5 ;
}
else
Statement 6 ;
}
```

Q 7 What is significance of break statement in switch case. Explain what happens when break statement is not given in switch.

Q 8 What is disadvantage of switch case statement when compared to if else statement.

CONTROL STRUCTURES
LOOPS

Loops are used for repeating certain set of statements. There are three types of loops in C. A) For loop , b) while loop c) do while loop

As an example let us consider the program for printing all numbers from 1 to 100.

```
/*program to print all numbers from 1 to 100 */
1 #include <stdio.h>
2 void main()
3 {
4   int  x;
5   for (x = 1 ; x <= 100 ; x++)
6    printf("%d\n" , x);
7 }
```

Line 5 and 6 form a for loop in which x varies from value 1 to 100 in the increments of 1 and for each value of x the printf statement is executed.

General syntax of for loop is

```
for ( initialization ; condition ; change)
{
 statement1;
 statement2;
}
```

Here condition is any boolean expression which will be evaluated to either true or false. statemant1 and statement2 are any valid C statement. Statement1 and statement2 are repeated as long as the condition is true. So the loop will terminate as soon as the condition becomes false.

Let us look at a few program that use for loop.

```
/*program to explain for loop */
1 #include <stdio.h>
2 void main()
3 {
4   int  x;
5   for (x = 100 ; x <= 1 ; x++)
6    printf("%d\n" , x);
7 }
```

The above program will not produce any output. This is because the counter x starts with 100. Then the condition is checked which is (x <= 1). But (100 <= 1) is false. Hence the loop terminates.

```
/*program to explain for loop */
1 #include <stdio.h>
2 void main()
3 {
4   int  x;
```

```
5   for (x = -1 ; x <= -100 ; x ++)
6    printf("%d\n" , x);
7 }
```

The above program will not produce any output. This is because the counter x starts with -1. Then the condition is checked which is (x <= 100). But (-1 <= -100) is false. Hence the loop terminates.

```
/*program to explain for loop */
1 #include <stdio.h>
2 void main()
3 {
4   int  x;
5   for (x = -100 ; x <= -1 ; x ++)
6    printf("%d\n" , x);
7 }
```

The above program will produce output from -100 to -1 on the screen. This is because the counter x starts with -100. Then the condition is checked which is (x <= -1). and (-100 <= -1) is true. Hence the loop executes and prints -100. Next time the value of x will increase by 1. Hence x will now become -99. And this process goes on.

Exercises:
State the output of following
a)

```
/*program to explain for loop */
1 #include <stdio.h>
2 void main()
3 {
4   int  x;
5   for (x = 1 ; x <= 100 ; x = x + 2)
6    printf("%d\n" , x);
7 }
```

b)

```
/*program to explain for loop */
1 #include <stdio.h>
2 void main()
3 {
4   int  x;
5   for (x = 100 ; x <= 100 ; x = x + 2)
6    printf("%d\n" , x);
7 }
```

c)

```
/*program to explain for loop */
1 #include <stdio.h>
2 void main()
3 {
4   int  x;
5   for (x = 100 ; x <= 200 ; x++)
6    printf("%d\n" , x + 2);
```

```
7 }
```

d)

```
/*program to explain for loop */
1 #include <stdio.h>
2 void main()
3 {
4   int  x , y;
5   for (x = 1 ; x <= 10 ; x++)
6   {
7     y = x % 2;
8     printf("%d %d\n" , x , y);
9   }
10 }
```

Following are several examples of programs which are solved using for loop.

Q1 : WAP to find sum of all numbers from 1 to 50.

```
/*program to find sum of all numbers from 1 to 50 */
1 #include <stdio.h>
2 void main()
3 {
4   int  x , s = 0;
5   for (x = 1 ; x <= 50 ; x++)
6     s = s + x;
7
8   printf( "sum = %d" , s);
9 }
```

Q2 : WAP to find sum of all numbers from 1 to n.
Q3: WAP to find sum of all numbers from m to n
Q 4: WAP to input any integer and print factorial of that integer.

```
#include <stdio.h>
void main()
{
  int  n ;
  long f = 1;

  // input the number n
   printf(" enter an integer ") ;
   scanf("%d" , &n);

 for (x = 1 ; x <= n ; x++)
   f = f * x;

 printf("factorial of number %d  is %ld " ,n,f);
}
```

Q5 : WAP to input an integer n and print it in reverse order.

```
#include <stdio.h>
void main()
{
  int  n , r;

  // input the number n
   printf(" enter an integer ") ;
   scanf("%d" , &n);

 for ( ;  n > 0 ; )
 {
   r = n % 10;
   printf("%d" , r) ;
   n = n / 10;
 }
}
```

Q6 : WAP to input an integer n and count the number of digits of that integer (Example if the number n = 1048 then output should be 4).

Q7 : WAP to input an integer n and find sum of digits of that integer(Example if the number n = 1048 then output should be 13).

Q8 : WAP to input an integer n and check if the number is Amstrong number.(A number is said to be Amstrong if sum of the cubes of digits is equal to the number itself).

```
#include <stdio.h>
void main()
{
  int  n , r , s = 0 , t;

  // input the number n
   printf(" enter an integer ") ;
   scanf("%d" , &n);
 t = n;
 for  ( ;  n > 0 ; )
 {
   r = n % 10;
   s = s + r * r* r;
   n = n / 10;
 }

 if ( s == t)
   printf( "number is amstrong" );
 else
   printf("number is not amstrong " );
}
```

Q9 : WAP to input integer n and check if it is Palindrome or not. (Eg if n = 121, it is Palindrome because number read from left or from right is same)

Q10: WAP to find sum of following series :
S = 1/1 + 1/2 + 1/3 + ------- + 1/n

```
#include <stdio.h>
void main()
{
  int  n , x;
 double s = 0;

  // input the number of terms
   printf(" enter number of terms ") ;
   scanf("%d" , &n);
   for (x = 1 ; x <= n ; x++)
     s = s + 1.0 /  x;

  printf("sum = %f " , s);
}
```

Q11: WAP to find sum of following series :
S = 1/2 + 3/4 + 5/6 + ------- + (add n such terms)

```
#include <stdio.h>
void main()
{
  int  n , x;
 double s = 0 , a = 1 , b = 2;

  // input the number of terms
   printf(" enter number of terms ") ;
   scanf("%d" , &n);
   for (x = 1 ; x <= n ; x++)
   {
     s = s + a / b;
     a = a + 2;
     b = b + 2;
   }
   printf("sum =  %f"  ,  s);
}
```

Q12: WAP to find sum of following series :
S = 3/2 + 6/4 + 9/6 + ------- + (add n such terms)

While loop

While loop is also used to repeat set of statements just like for loop. Syntax of while loop is as follows

```
while (condition)
{
 Statement1;
 Statement2;
 "   "
 "   "
```

}

Statements in the while loop will be repeated as long as the condition is true. Following eg shows output from 1 to 10.

```
1 int x = 1;
2 while ( x <= 10)
3 {
4   printf("%d\n" ,  x) ;
5   x++;
6 }
```

Q 1 State the output of following

#include <stdio.h>
void main()
{
int x = 1;
while (x <= 10)
printf("%d\n" , x);
x++;
}
	#include <stdio.h>
	void main()
	{
	int x = 1;
	while (x <= 10)
	{
	printf("%d\n" , x);
	x++;
	}
	}
		#include <stdio.h>
		void main()
		{
		int x = 0;
		while (x <= 20)
		{
		x++;
		printf("%d **\n" , x);
		}
		}

Q 2 Write a program to input integer n and print it in reverse order.

```
#include <stdio.h>
void main()
{
 int n , r ;
 printf("enter a number ") ;
 scanf("%d" ,  &n);
 while ( n > 0)
 {
  r = n % 10;
  printf("%d" ,  r) ;
  n = n % 10;
 } // end while
} //end main
```

Q 3 Write a program which follows following algorithm.

Step 1 : Input integer n

Step 2 : if n is odd then multiply n by 3 and add 1 to it otherwise if n is even divide n by 2.

Step 3 : go back to step 2 if n is not equal to 1. Other wise if n is 1 stop.

The above algorithm is used for converging integer n to 1. For example is n = 5, then algorithm will converge n to 1 as follows.

N = 5 here n is odd hence n = n * 3 + 1 ie n = 5 * 3 + 1 ie n = 16

N = 16 here n is even hence n = n / 2 ie n = 16/2 ie n = 8

N = 8 here n is even hence n = n / 2 ie n = 8/2 ie n = 4

N = 4 here n is even hence n = n / 2 ie n = 4/2 ie n = 2

N = 2 here n is even hence n = n / 2 ie n = 2/2 ie n = 1

Q 4 Write a program find factorial of integer n using while loop.

Q 5 Write a program to find sum of following series
S = 1/2 + 3/4 + 5/6 + ------- + (add n such terms)

do while loop

Syntax of do while loop

do
{
Statement 1;
Statement 2;
" "
" "
}
while (condition);
	The statements inside do while loop will repeat as long as the condition is true.

Example 1

int x = 1;
do
{
printf(%d\n" , x);
x++;
}
while (x <= 10);
	The output is all numbers from 1 to 10

Example 2

int x = 11;
do
{
printf("%d\n" , x);
x++;
}
while (x <= 10);
	The output is 11.
	This shows that the do while loop executes atleast once.
	x is initially 11. The do while loop will execute once and print 11 after which x becomes 12 (due to x++). At this point of time the condition x <= 10 becomes false and the loop breaks

Q 1 State the difference between while and do while loop.

while loop may not execute even once if the condition is initially false. For example the following while loop will not execute at all.
 	
int x = 11;
while (x <= 10)
{
printf("%d\n" , x);
x++;
}
	do while loop will execute atleast once irrespective of the condition. This is because in do while loop condition is written at the bottom of loop and hence the statements execute once and then the condition is checked. Following example will print 11 although the condition in do while is false
	
	int x = 11;
	do
	{
	printf("%d\n" , x);
	x++;
	}
	while (x <= 10);