UNIT-5

PROGRAMMING IN C

Introduction of C:

- ʻCʹ seems a strange name for a programming language. But this strange sounding language is one of the most popular computer languages today because it is a structured, high-level, machine independent language. It allows software developers to develop programs without worrying about the hardware platforms where they will be implemented.
- ʻCʹ language was developed by Dennis Ritchie at Bell laboratories in 1972.

Some facts about C:

- In 1988, the American National Standards Institute (ANSI) had formalized the C language.
- C was invented to write UNIX operating system.
- C is a successor of 'Basic Combined Programming Language' (BCPL) called B language.
- Linux OS, PHP, and MySQL are written in C.
- C has been written in assembly language.

C has been popular for the following reasons:

- One of the early programming languages.
- Still, the best programming language to learn quickly.
- C language is reliable, simple and easy to use.
- C language is a structured language.
- Modern programming concepts are based on C.
- It can be compiled on a variety of computer platforms.

- Universities preferred to add C programming in their courseware.

Advantages of C:

- C is the building block for many other programming languages.
- Programs written in C are highly portable.
- Several standard functions are there (like in-built) that can be used to develop programs.
- C programs are collections of C library functions, and it's also easy to add own functions to the C library.
- The modular structure makes code debugging, maintenance and testing easier.

Disadvantages of C:

- C does not provide Object Oriented Programming (OOP) concepts.
- There are no concepts of Namespace in C.
- C does not provide binding or wrapping up of data in a single unit.
- C does not provide Constructor and Destructor.

Character set used in C:

A character denotes any alphabets, digit or special symbol used to represent information.

Letters	**Digits**	**Special Characters**
Upper case A....Z Lower case a......z	All decimal digits 0..............9	, . ; ? ( ) {} $ % *

➢ **Use of comments**: Comment lines are not executing statement.

//Program written

- **Identifiers:**

Identifiers refer to the name of variables, functions. Both upper case and lower case letters are permitted, through lower case letters are commonly used.

- **Keywords:**

Keywords are the words whose meaning has already been explained to the C compiler. All keywords must be written in lower case. There are **32** keywords in C. Keywords cannot be used as variable name.

Auto	double	Int	struct
break	else	Long	switch
Case	enum	register	typedef
Char	extern	return	union
Const	float	Short	unsigned
Signed	continue	For	void
Default	goto	sizeof	volatile
Do	if	static	while

- **Token:**

The smallest individual unit in a program is called C token. C token has six tokens.

Keywords	**Constant**	**String**	**Operators**	**Special Symbol**	**Identifier**

int, float	3.14, 10	“ABC” “Kajol”	+ - * /	{ } ( ) [ ]	a, b, l, h

- **Data type in C language:**

Data types are mainly used to define the type and nature of data such that compiler detects and proceeds.

Data types used in C are as follows:

i) int: - int is the keyword for integer. It contains the whole numbers between -32,768 to 32,767. It requires 2 bytes memory and its type specifier is %d.

Examples: - int a=10;

ii) float: - float is the keyword for floating numbers i.e. fractional numbers. It contains number $3.4e^{-38}$ to $3.4e^{+38}$. It requires 4 bytes memory and its type specifier is %f. Example: - float pi=3.14;

iii) char: - char is the keyword for character. It represents the single alphabet. It requires 1-byte memory and its type specifier is %c. Example: - char choice;

- **Variable**: -

A variable is a data name that may be used to store a data value. A variable may take different values at different times during program execution. Variable names may consist of letters, digits and the underscore (_) characters. A variable name can be chosen by the programmer in a meaningful way so as to reflect its functions or nature in the program.

Rules for variable naming: -

1. They must begin with a letter.

2. ANSI (American National Standard Institute) standard recognizes a length of 31 characters. However, length should not be normally more than 8 characters, since only the first 8 characters are treated as significant by many compilers.

3. Upper case and lower case are significant that is, the variable **area** is not the same as **Area** or **AREA**.

4. It should not be a keyword.

5. White space is not allowed.

Example: - Roll No is invalid but Roll_No is valid variable name.

Types of variable: -

There are 2 types of variables.

- **Numeric Variable**: - The variables which stores the numeric data only is called numeric variable. It can be whole number or decimal number. Example: 10, 11, 3.14 etc.
- **String Variables**: - The variable which stores the character data only is called String variable. It can be a single character or string. Examples: ′a′, ″apple″ etc.

➢ **<u>Constant</u>**: -

A constant is a value that does not change during the program execution. There are several types of constants. They are:

a) **Integer constant**: - An integer constant refers a sequence of digits. Examples: 123, 23, 0 etc.
b) **Real (Floating constant)**: - Floating point constant contains a decimal point. Examples: 3.14, 6.12, etc
c) **Character**: - A character constants are enclosed within single quote. Examples: ′a′, ′b′ etc.
d) **String constant**: - A string constant is a sequence of characters enclosed in double quote.
   Examples: ″Hi this is programming language″, ″2015″ etc.

e) **Symbolic constant**: - A symbolic constant is simply an identifier used in place of a constant. In C symbolic constants are usually defined at the start of program. Examples:

```
# define   pi   3.14
```

<u>OPERATOR</u>

An operator is a symbol that instructs C to perform some operation on one or more operands. The data on which operator are performed is called Operand. For example: p=3+4; P, 3 and 4 are operands whereas = and + are operator and p=7 is the result.

The operators provided by C language are as follows:

1. Arithmetic Operator:

Operators which are used in mathematical expression are called arithmetical operators. The various arithmetic operators are:

Operators	**Name**	**Example**
+	Addition	a=2+3=5
-	Subtraction	a=5-1=4
*	Multiplication	a=3*4=12
/	Division	a=12/2=6
%	Modulus(Remainder)	a=12%2=0

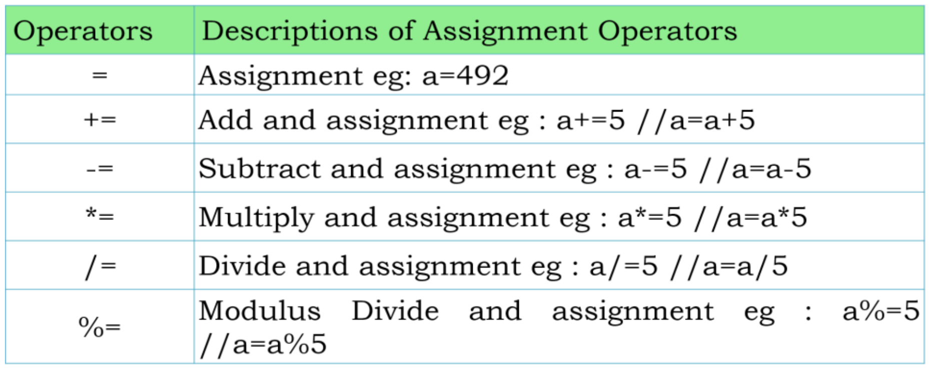
2. Logical Operator

These operators are generally used in conditional expression. The three logical operators in C are as follows:

a) && (Logical AND)
b) || (Logical OR)
c) !(Logical NOT)

3. Assignment Operator

The assignment operator calculates the expression on the right side and gives the values of left side variables.



Operators	Descriptions of Assignment Operators
=	Assignment eg: a=492
+=	Add and assignment eg : a+=5 //a=a+5
-=	Subtract and assignment eg : a-=5 //a=a-5
*=	Multiply and assignment eg : a*=5 //a=a*5
/=	Divide and assignment eg : a/=5 //a=a/5
%=	Modulus Divide and assignment eg : a%=5 //a=a%5

Here, at first the arithmetic operation is performed than only assignment operation is performed.

4. Relational Operator

It is used to compare the values between operands and gives a result whether it is true or false. It is also called comparison operator due to its comparing characteristics. There are 6 relational operators are:

<	Less than
< =	Less than equal to
>	Greater than
>=	Greater than equal to
= =	Equal to
! =	Not equal to

5. Ternary Operator (? :)

Conditional operator is also known as ternary operator helps to check the Boolean values that true or false.

Syntax:

<condition>? <expression-1>: <expression-2>

If the condition is true, expression-1 will be executed; otherwise, expression-2 will be executed. The ? is similar to if - else statement.

Example:

```
int x, a, b;

a=6;

b=10;

x= (a<b)? a:b;

printf("%d",x);
```

Here, the value of x is 6.

6. Unary Operator

The increment and decrement operator are very useful in C language. The syntax of operator is:

a. + + variable name (increment prefix) eg: ++a
b. variable name + + (increment postfix) eg: a++
c. -- variable name (decrement prefix) eg: --a

d. variable name- - (decrement postfix) e.g: a--

7. Comma Operator

The comma operator can be used to link the related expressions together. A comma-linked list of expressions is evaluated left to right and the value of right-most expression is the value of the combined expression. For examples: The statement

F= (a=2, b=3, a+b)

First assigns the value 2 to **a**, then assign 3 to **b** and finally assigns 5 to **F**.

Header file:

A header file is a file with extension **.h** which contains C function declarations and macro definitions to be shared between several source files.

#include<stdio.h>

It is the standard input/output header file. It contains the function definition of input output functions such as scanf(), printf(), etc.

#include<conio.h>

It is a header file use in console input output function such as clrscr() , getch(), etc.

#include<math.h>

This header file is used for mathematical function such as pow(), sqrt(), sin(), etc.

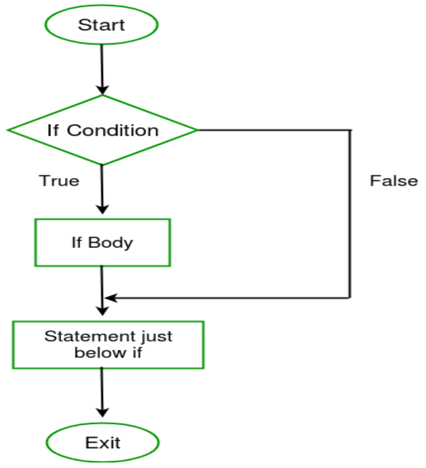
#include<string.h>

This header file is used for string processing such as strlen(), strcpy, etc.

#DECISIONS MAKING STATEMENTS :

The statement which display one statement when the condition is true, otherwise display another statement is known as decision-making statement. Since these statement "control" the flow of execution, they are also known as control statements.

C language has following types of decision-making statements are available as follows:



a) Simple if statement

b) if –else statement

c) Nested if-else statement

d) else---if ladder

e) Switch –case statement

a) Simple if Statement

Syntax:

```
if(condition)
{
Statement-block;
}
```

The ʹstatement-blockʹ may be a single statement or a group of statements. 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. Remember, when the condition is true both the statement block and the statement-x are executed in sequence.

Consider the following segment of a program that is written for processing of marks obtained in an entrance examination.

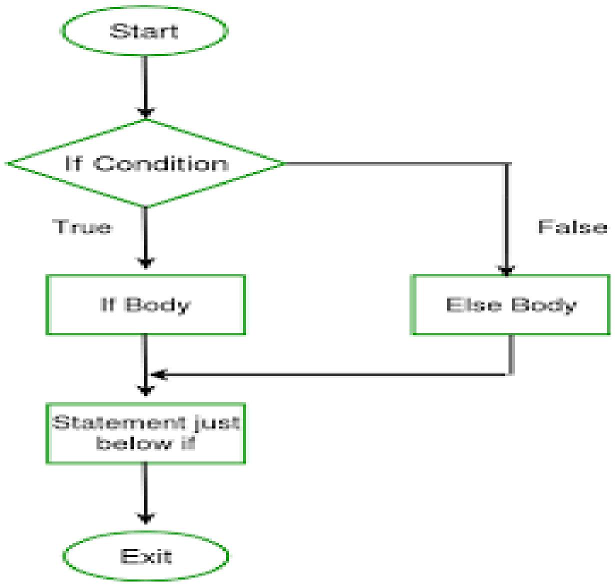
```
......................



......................

if (category= = sports)

{

Marks= marks + bonus_marks;

}

printf(ʺ %f ʺ,marks);

......................

......................
```

The program tests the type of category of the student. If the student belongs to the SPORTS category, then additional bonus_marks are added to his marks before they are printed. For others, bonus_marks are not added.

b) if---else statement

The if --- else statement is an extension of the simple if statement.

If the test expression is true, then the true-block statements, immediately following the if statement are executed; otherwise, the false-block statements are executed. In either case, either true-block or false block will be executed, not both. This is shown in flowchart below. In both the cases, the control is transferred subsequently to the statement-x.



if (test expression)

{

True –block statement(s);

}

```
else
{
False-block statement(s);
}
Statement-x;
```

Let us consider an example of counting the number of boys and girls in a class. We use code 1 for a boy and 2 for a girl. The program statement to do this may be written as follows.

```
......................
......................
if(code=                    =                    1)
boy= boy+1;
if(code= = 2)
girl=girl+1;
......................
......................
```

The first test determines whether or not the student is a boy. If yes, the number of boys is increased by 1 and the program continues to the second test. The second test again determines whether the student is a girl. This is unnecessary. Once a student is identified as a boy, there is no need to test again for a girl, not both. The above program segment can be modified using the else clause as follows.

```
......................
```

……………………

if(code = = 1)

boy=boy+1;

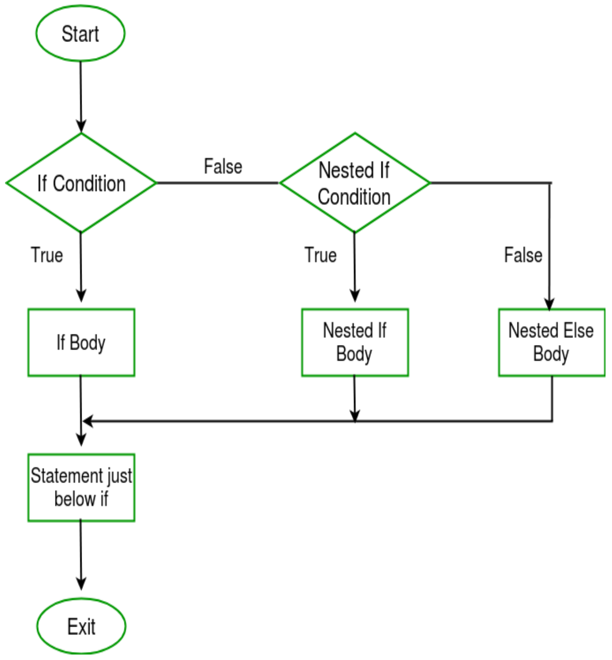
else

girl=girl+1;

……………………

c) Nested if ----- else statement

When a series of decisions are involved, we may have to use more than one if ----else statement in nested form as shown below:



if(test condition-1)

{

if(test condition-2)

{

Statement-1;

}

else

{

```
Statement-2;
}
}
else
{
Statement-3;
}
Statement-x;
```

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 executed; otherwise the statement-2 will be executed and then the control is transformed to the statement –x.

A commercial bank has introduced an incentive policy of giving bonus to all its deposit holders. The policy is as follows: A bonus of 2% of the balance held on 31st December is given to everyone, irrespective of their balance, and 5% is given to female account holders if their balance is more than Rs. 5000. This logic can be coded as follows:

```
.......
if(sex is female)
{
if(balance >5000)
bonus=0.05*balance;
```

else

bonus=0.02*balance;

}

else

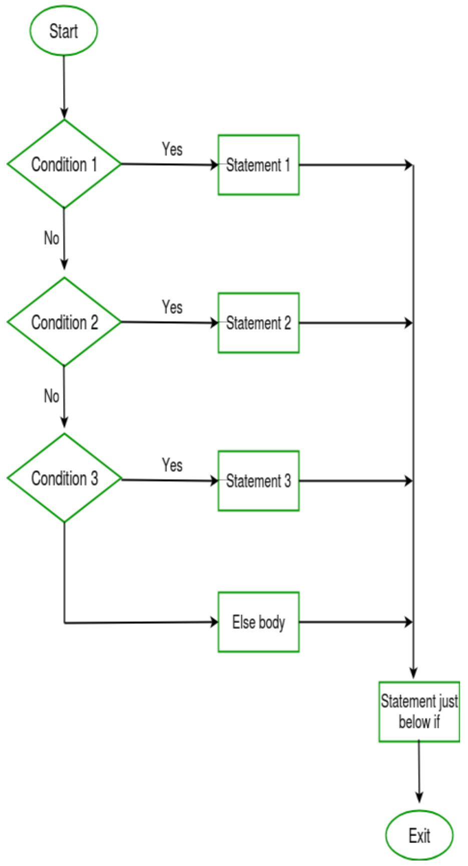
{

bonus=0.02*balance;

}

balance=balance+bonus;

....



d) The else---if ladder

This is another way of putting ifs together when multipath decisions are involved. A multipath decision is a chain of ifs in which the statement associated with each **else** is an **if**.

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) to 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 ladder). When all the "n" conditions become false, then the final else containing the default – statement will be executed.

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

Average Marks	**Grade**

80 to 100	Distinction
60 to 79	First Division
50 to 59	Second Division
40 to 49	Third Division
0 to 39	Fail

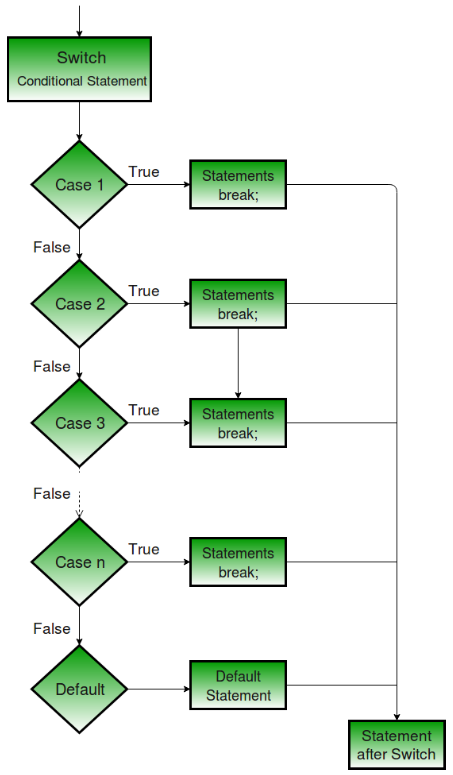
This grading can be done using the else if ladder as follows:

```
if(marks>79)
        grade=″Distinction″;
else if(marks>59)
        grade=″First Division″;
else if(marks>49)
        grade=″Second Division″;
else if(marks>39)
        grade=″Third Division″;
else
        grade=″Fail″;
printf(“%s\n″,grade);
```

e) Switch- Case Statement

The switch statement tests the value of a given variable (or expression) against a list of case values and when a match is found, a block of statements associated with that case is executed. The general form of the switch statement is as shown below:

```
switch(expression)
{
case value-1:
block-1;break;
case value-2:
block-2;
break;
.........................
.........................
default:
default – block;
break;
}
```



statement – x;

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 follow the case are executed.

The **break** statement at the end of each block signals the end of a particular case and causes an exit from the switch statement, transferring the control to the statement-x following with switch.

The **default** is an optional case. When present, it will be executed if the value of 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.

Let us consider an example, which displays the name of 7 days according to input numbers. This can be shown as follows:

```
switch(ch)

{

case 1:

printf("Sunday");

break;

case 2:

printf("Monday");

break;
--------------
--------------
case 7:
printf(“Saturday”);
defaults:
printf(“Enter the number between 1 to 7”);
}
```

Looping:

The process of executing the same statement repeatedly until a condition is satisfied is called looping.

If the task or set of instructions required to be executed “n” number of times, we can use loop statements.

In C- language we have 3 types of looping structures.

1. For loop

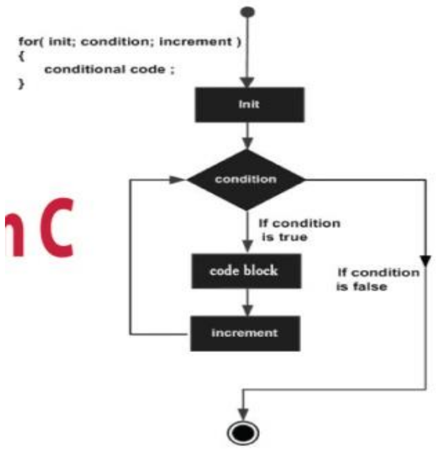
2. While loop
3. Do-while loop

1. For Loop

The for loop is applied in the situation when you exactly know how many times you want to execute the statements.

Syntax of for loop:

```
for(initialization; condition; increment/decrement)
{
--------------------------
--------------------------
}
```



```
for( init; condition; increment )
{
    conditional code ;
}
```

Example:

```
 #include<stdio.h>
#include<conio.h>
main( )
{
      int i;
      for(i=1;i<=10;i++)
      {
                  printf("I am a student of calss 12\n");
      }
getch ( );
}
```

2. While Loop

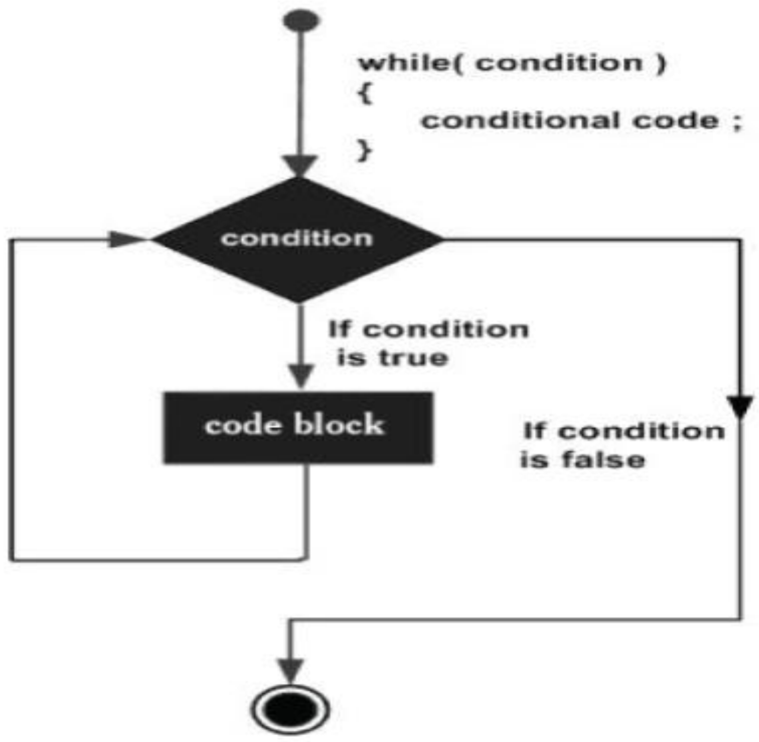
The while loop executes a statement or a block of statements as long as a condition evaluates to true. The while loop is mainly used in situations where you don't know in advance how many times the loop will be executed. The loop terminates when the condition evaluates to false.

Syntax of while loop

```
Initializations;
while(condition)
{
Statements
..........................
Increment/decrement
}
```

Example of While -Loop:

```
#include<stdio.h>

#include<conio.h>

main( )

{

    int i=1;

    while(i<=10)

    {

              printf("I am a student of calss 12\n");

          i++;

   }
```

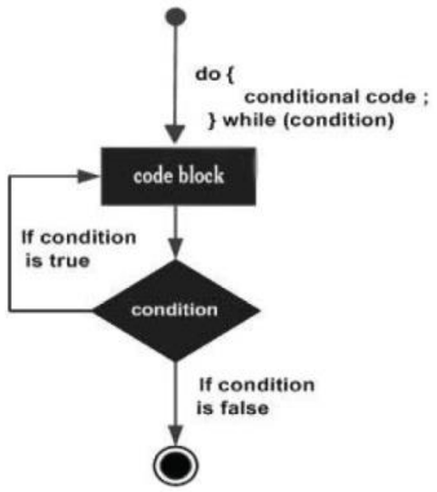


getch( );

}

3. do-while loop

The working of do-while loop is identical to that the while loop, except that in the do-while loop, a condition is checked at the end of the loops after each iteration. It means that the do-while loop definitely executes at least once even if the condition is false. Unlike the while loop, a semicolon has to be placed after while expression in the do-while loop; otherwise, the loop will not execute.



<u>Syntax of do-while loop:</u>

```
initialization;
do
{
Statements;
Increment/decrement
}
while condition ( );
```

Examples of do-while loop

```
#include<stdio.h>

#include<conio.h>

main()

{

    int i=1;

    do

    {

                printf(" I am a student of calss 12\n");
```

```
                        i++;

      }while(i<=5);

getch();

}
```

<u>Nested loop</u>

The loop inside the another loop is called nested loop.

Syntax:

```
for(initialization; condition; increment/decrement)

{

for(initialization; condition; increment/decrement)

{

Statement;

}

}
```

<u># Differences between while and do-while loop:</u>

While loop	**do-while loop**
1. Condition is evaluated before the statement is executed.	1. Condition is evaluated after the statement is executed.
2. It is an entry control loop.	2. It is an exit control loop.
3. It has a keyword While.	3. It has two keyword do and while.
4. Semicolon is not placed after the while condition.	4. Semicolon is placed after the while condition.
5. It does not execute a statement when the condition is false.	5. It executes statement once even the condition is false.

6. Example:
#include<stdio.h>
#include<conio.h>
main( )
{
	6. Example:
	#include<stdio.h>
	#include<conio.h>
	main( )
	{

int i=1;
while(i<=10)
{
printf("I am a student of class 11\n");
i++;
}
getch( );
}
	int i=1;
	do
	{
	printf(" I am a student of class 11\n");
	i++;
	}while(i<=5);
	getch( );
	}

#Loop Interruption

To interrupt the normal way of program, there are two statements for the loop interruption.

Break and continue statement:

1. Break statement:

The break statement is used to exit from a while, for, do-while or switch structures. It can only be used inside the body of a for, while, do-while, or switch statement.

2. Continue Statement:

The continuous causes the loop to be conditioned with the next iteration after skipping any statements in between. The continue statements tells the compiler ″skip the following statements and continue with the next iteration″.

# Differentiate between Break and Continue Statement

Break Statement	**Continue Statement**
1.The break statement is used to terminate the control from the switch-case and loop structure.	1.The continue statement is used to by-pass the execution of the further statements.
2. When break statement encountered the entire loop or switch statement is terminated.	2. When continue statement is encountered the entire loop is not terminated; only that particular iteration is skipped.
3. It uses a keyword break.	3. It uses a keyword continue.

4. It can be used in both loop and switch case.	4. It can be only used in loop structure.

5. Example:
main( )
{
int i;
for (i=1;i<=10;i++)
{
if(i= =2)
break;
printf("%d",i);
}
getch ( );
}
Output: 1
	5. example:
	main( )
	{
	int i;
	for (i=1;i<=10;i++)
	{if(i= =2)
	continue;
	printf("%d\t",i);
	}
	getch( );
	}
	Output:
	1 3 4 5 6 7 8 9 10

<u>Homework Questions</u>

<u>C-1 (Introduction of C)</u>

1.1 Define data type. Explain different types of data types used in C programming with examples.

1.2 What is an operator? Explain different types of operators used in C programming with examples.

1.3 What is variable? Write its types. Define the terms identifier and keywords with example.

1.4 Differentiate between while and do-while loop with examples.

1.5 What is looping? Compare ʺForʺ, ʺwhileʺ,ʺdo-whileʺ loops with examples.

1.6 Differentiate between break and continue statements with examples.

THE END

PROGRAMMING PART-1

1. WAP to find out whether the input number is even or odd?

```
#include<stdio.h>
#include<conio.h>
Void main()
{
int n;
printf("Enter any number=");
scanf("%d",&n);
if(n%2==0)
printf("The entered number is even.");
else
printf("The entered number is odd.");
getch( );
}
```

2. WAP to accept any 3 numbers and print the largest number among them?

```
#include<stdio.h>
#include<conio.h>
void main( )
{
```

```
int a,b,c;
printf("Enter any three  numbers=");
scanf("%d%d%d",&a,&b,&c);
if(a>b&&a>c)
printf("The largest integer=%d",a);
else if(b>a&&b>c)
printf("The largest integer=%d",b);
else
printf("The largest integer=%d",c);
getch( );
}
```

3. WAP that checks whether the number entered by user is exactly divisible by 5 but not by 11?

```
#include<stdio.h>
#include<conio.h>
Void main( )
{
int n;
printf("Enter a number =");
scanf("%d",&n);
if(n%5==0&&n%11!=0)
printf("Entered number is divisible by 5 but not by 11″);
else
printf("Exit.");
getch( );
}
```

4. WAP that enter SP and CP and determine whether there is profit of loss?

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

```
main( )
{
int sp,cp,p,l;
printf("Enter the selling price and cost price=");
scanf("%d%d",&sp,&cp);
if(sp>cp)
{
p=sp-cp;
printf("Profit=%d",p);
}
else
{
l=cp-sp;
printf("Loss=%d",l);
}
getch( );
}
```

5. WAP to display the series up to 10th terms.

1 5 9 13 …….

```
#include<stdio.h>
#include<conio.h>
main( )
{
int a=1,c;
for(c=1;c<=10;c++)
{
printf("%d\t",a);
a=a+4;
}
getch( );
}
```

```
#include<stdio.h>
#include<conio.h>
main( )
{
int n,i,sum=0;
printf("Enter a number=");
scanf("%d",&n);
for(i=1;i<=n;i++)
{
if(i%2==0)
sum=sum+i;
}
printf("Sum of even numbers up to %d = %d", n,sum);
getch();
}
```

6. WAP to display the sum of “n” terms of even numbers?

7. WAP to display the multiplication of “n” number?

```
#include<stdio.h>
#include<conio.h>
main( )
{
int n, i, ans;
printf("Enter any numbers=");
scanf("%d",&n);
for(i=1; i<=10; i++)
{
ans=n*i;
printf("%d*%d=%d\n",n,i,ans);
}
getch( );
```

8. WAP to print the factorial of a given number?

```
#include<stdio.h>

#include<conio.h>

main( )

{

int n,i,f=1;

printf("enter the number=");

scanf("%d",&n);

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

printf("Factorial of %d =%d",n,f);

getch( );

}
```

9. WAP to print the 10 positive integers and their factorials?

```
#include<stdio.h>
#include<conio.h>
main( )
{
int n,i,f=1;
for(i=1;i<=10;i++)
{
f=f*i;
printf("Factorial of %d =%d\n",i,f);
}
getch( );
}
```

10. WAP to check whether the input number is prime number or not?

```
#include<stdio.h>
#include<conio.h>
main( )
{
int i, n;
printf("\nEnter any number : ");
scanf("%d",&n);
for(i=2;i<n;i++)
{
if(n%i==0)
{
printf("\nThe number %d is not prime",n);
break;
}
}
if(i==n)
printf("\nThe number %d is prime",n);
getch( );
}
```

11. WAP to display the prime numbers from 1 to 100?

```
#include<stdio.h>
#include<conio.h>
main ( )
{
int n,i;
for(i=1;i<=100;i++)
{
 for(n=2;n<i;n++)
 {
 if(i%n==0)
 break;
 }
 if(i==n)
 printf("%d\t",n);
}
 getch();
 }
```

12. WAP to display Fibonacci series i.e. 0   1   1   2   3   5 ………

```
#include<stdio.h>
#include<conio.h>
main( )
{
int a=0,b=1,c,i;
printf("%d\t%d\t",a,b);

for(i=1;i<=10;i++)
{
c=a+b;
a=b;
b=c;
printf("%d\t",c);
}
getch();
}
```

13. WAP to display 1 for Sunday, 2 for Monday etc.

```
#include<stdio.h>
#include<conio.h>
main ( )
{
int choice;
printf("Enter the numbers of the days=");
scanf("%d",&choice);
switch(choice)
{
case 1:
printf("Sunday");
break;
case 2:
printf("Monday");
break;
case 3:
printf("Tuesday");
break;
case 4:
printf("Wednesday");
break;
case 5:
printf("Thursday");
break;
case 6:
printf("Friday");
break;
case 7:
printf("Saturday");
break;
default:
printf("Wrong Choice");
}
getch( );
}
```

14. WAP to read a 4-digit number and display it in reverse order?

```
#include<stdio.h>
#include<conio.h>
main()
{
int num;
printf("enter a number=");
scanf("%d",&num);
printf("The number in reverse order=");
printf("%d",num%10);
num=num/10;
printf("%d",num%10);
num=num/10;
printf("%d",num%10);
num=num/10;
printf("%d",num%10);
getch( );
}
```

15. WAP to display 1

12

123

1234

12345 ?

```
#include<stdio.h>

#include<conio.h>

main()
```

```
{

int i,j;
for(i=1;i<=5;i++)
{
for(j=1;j<=i;j++)
{
printf("%d",j);
}
printf("\n");
}
getch( );
}
```

16. WAP to display

```
1
22
333
4444
55555   ?
```

```
#include<stdio.h>

#include<conio.h>

main ( )

{

int i,j;

for(i=1;i<=5;i++)

{

for(j=1;j<=i;j++)

{

printf("%d",i);

}

printf("\n");

}

getch( );

}
```

17. WAP to display

```
*

**

***

****
```

*******   ?**

```
#include<stdio.h>

#include<conio.h>

main()

{

int i,j;

for(i=1;i<=5;i++)

{

for(j=1;j<=i;j++)

{

printf("*");

}

printf("\n");

}

getch( );

}
```

18. WAP to display   4321

321

21

1 ?

```
#include<stdio.h>
#include<conio.h>
main ( )
{
int i,j;
for(i=4;i>0;i--)
{
for(j=i;j>0;j--)
{
printf("%d",j);
}
printf("\n");
printf(" ");
}
getch( );
}
```

19. WAP to display **12345**

1234

123

12

1 ?

```
#include<stdio.h>

#include<conio.h>

main()

{

int i,j;

for(i=5;i>=1;i--)

{

for(j=1;j<=i;j++)

{

printf("%d",j);
}
printf("\n");
}
getch();
}
```

20. WAP to display 55555

4444

333

22

1 ?

```
#include<stdio.h>
#include<conio.h>
main( )
{
int i,j;
for(i=5;i>=1;i--)
{
for(j=1;j<=i;j++)
{
printf("%d",i);
}
printf("\n");
}
getch( );
}
```

21. WAP to display

```
*****
****
***
**
*   ?
```

```
#include<stdio.h>
#include<conio.h>
main( )
{
int i,j;
for(i=5;i>=1;i--)
{
for(j=1;j<=i;j++)
{
printf("*");
}
printf("\n");
}
getch( );          }
```

22. WAP to enter a string and check whether the entered string is palindrome or not? [A string is said to be palindrome if it remains same if we reverse it. Eg:- ADA,LIRIL,MALAYALAM,MADAM]

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char str[30],temp[30];
printf("Enter a string\n");
scanf("%s",str);
strcpy(temp,str);
strrev(temp);

if(strcmp(temp,str)==0)

printf("String is palindrome");

else

printf("String is not palindrome");

getch();

}
```

23. WAP to display the Armstrong number to ″n″ numbers. [An Armstrong number is that whose sum of cube of digits is equal to number itself. Example:-371=$3^3+7^3+1^3$]

```
#include<stdio.h>

#include<conio.h>

main()

{

int a,j,arm,sum,k;

printf("Armstrong numbers up to what?");
```

```
scanf("%d",&arm);

for(a=1;a<=arm;a++)

{

sum=0;

j=a;

while(j!=0)

{

k=j%10;

sum=sum+k*k*k;

j/=10;

}

if(a==sum)

printf("\n%d is Armstrong",sum);

}

getch();
```

```
}
```

24. **WAP a menu driven program to perform the following task**

a. to find the factorial of a given number

b. to find whether the input number is prime or not.

c. to find whether the input number is odd or even.

d. Multiplication table of input number

e. exit

```
#include<stdio.h>

#include<conio.h>

#include<stdlib.h>

main()

{

int choice,num,i;

long int m=1;

printf("\n\n1.Factorial number\n");
printf("2. prime\n");

printf("3.Odd or even\n");

printf("4.Multiplication table\n");

printf("5.Exit\n");

printf("Enter your choice(1/2/3/4/5)=");

scanf("%d",&choice);

switch(choice)

{

case 1:


printf("enter the number=");

scanf("%d",&num);

for(i=num;i>0;--i)

m=m*i;

printf("Factorial of %d =%ld",num,m);

break;
```

```
case 2: printf("Enter number=");

scanf("%d",&num); for(i=2;i<num;i++)

{ if(num%i==0)

{ printf("\n Not prime.");

break;

}

}

if(i==num)

printf("\n Prime number");

break;

case 3:


printf("Enter a number=");

scanf("%d",&num);

if(num%2==0)

printf("\n Even number");
```

```
else printf("\n Odd numbers");
break;
case 4:

printf("Enter number=");
scanf("%d",&num);
for(i=1;i<=10;i++)
{ int mult=num*i;
printf("\t %dX%d=%d\n",num,i,mult);
} break;
case 5:

exit(0);
default:
printf("You are not allowed to type other.");
}
```

```
getch();

 }
```

25. WAP to accept two numbers and performs Addition, subtraction, multiplication and division?

```
#include<stdio.h>

#include<conio.h>

main()

{ char choice;

int n1,n2,result;

printf("Enter the sign");

scanf("%c",&choice);

printf("Enter any two numbers\n");

scanf("%d%d",&n1,&n2);

switch(choice)

{

case'+':

result=n1+n2;
```

```
printf("sum=%d",result);

break;

case'-':

result=n1-n2;

printf("subtraction=%d",result);

break;

case'*':

result=n1*n2;

printf("Multiplication=%d",result);

break;

case'/':

result=n1/n2;

printf("Division=%d",result);

break;

case'%':

result=n1%n2;

printf("Remainder=%d",result);

break;
```

```
default:

printf("Enter the valid key.");

}

getch();

}
```

26. WAP to make a report card of exam?

```
#include<stdio.h>

#include<conio.h>

main()

{ char name[30];

int clas,eng,nep,phy,chem,cmp,total;

float per;

printf("\nEnter the name of a student=");

scanf("%s",name);

printf("\n Enter the class=");

scanf("%d",&clas);

printf("\n Enter the marks in English, Nepali, Physics, Chemistry, Computer=");
```

```
scanf("%d%d%d%d%d",&eng,&nep,&phy,&chem,&cmp);

total=eng+nep+phy+chem+cmp;

per=total/5;

printf("NAME=%s\n",name);

printf("CLASS=%d\n",clas);

printf("TOTAL=%d\n",total);

printf("Percentage=%0.2f\n",per);

{

if(eng>=35&&nep>=35&&phy>=35&&chem>=35&&cmp>=35)

printf("Result=Passed\n");

else

printf("Result=Failed\n");

if(per>=75)

printf("Division=Distinction");

else if(per>=60&&per<75)

printf("Division=First");

else if(per>=45&&per<60)

printf("Division=Second");
else if(per>=35&&per<45)
printf("Division=Thid");
else printf("Division=Fail");
}
getch();
}
```

THE END

ARRAY

Definition: A composite variable capable of holding multiple data of same data type under a single variable name is called array.

Declaration of Array:

datatype array_name [size];

Example: - int marks[10]; int x[4]={1,2,3,4};

- Array size is fixed at the time of array declaration.
- Generally, we use for loop () to assign data or item in memory and also use this loop to extract data from memory location.

Types of Array:

There are 2 types of array.

a) One Dimensional Array (1-D)

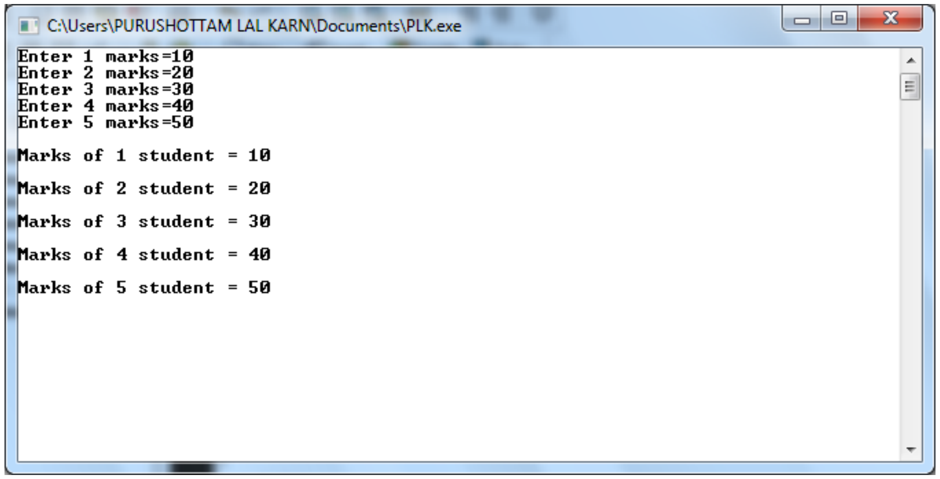
b) Two Dimensional Array (2-D)

One dimensional Array(1-D)	**Two dimensional Array(2-D)**
1) In 1-D array only one size is used.	1) In 2-D array two sizes are used.
2) It is called list also.	2) It is called table or matrix.
3) Declaration: datatype array_name[size];	3) Declaration: datatype array_name[size1][size2];
4) Eg:- int mark [20];	4) Eg:- float sale[2][3];
5) In 1-D array size denotes either row or column.	5) In 2-D array size 1 denotes row and size2 denotes column.
6) One looping statement is used to create or display array items or data.	6) Two looping statements are used to create or display array items or data.

A1. WAP that inputs the marks of 5 students and display on the screen?

```
#include<stdio.h>

#include<conio.h>

main( )

{

int marks[5],i;

for(i=0;i<5;i++)

{

printf("Enter %d marks=",i+1);

scanf("%d",&marks[i]);

}

for(i=0;i<5;i++)

{

printf("\nMarks of %d student = %d \n",i+1,marks[i]);

}

getch( );

}
```

OUTPUT:



C:\Users\PURUSHOTTAM LAL KARN\Documents\PLK.exe

```
Enter 1 marks=10
Enter 2 marks=20
Enter 3 marks=30
Enter 4 marks=40
Enter 5 marks=50

Marks of 1 student = 10

Marks of 2 student = 20

Marks of 3 student = 30

Marks of 4 student = 40

Marks of 5 student = 50
```

A2. WAP to store 10 numbers and print the largest among them?

```
#include<stdio.h>

#include<conio.h>

main( )

{

int n[10],i,largest;

for(i=0;i<10;i++)
{
printf("Enter any %d numbers=",i+1);
scanf("%d",&n[i]);
}
largest=n[0];
for(i=0;i<10;i++)
{
if(n[i]>largest)
{
largest=n[i];
}
}
printf("Largest Number =%d.",largest);
getch( );
}
```

A3. WAP to find the largest number among "nth" numbers?

```
#include<stdio.h>

#include<conio.h>

main()

{

int i,n,num[100],max;

printf("Enter the size of array not more than 100\n");

scanf("%d",&n);

for(i=0;i<n;i++)

{

printf("Enter the %d number =",i+1);
scanf("%d",&num[i]);

}

max=num[0];

for(i=1;i<n;i++)

{

if(num[i]>max)

max=num[i];

}

printf("The largest number in array = %d ", max);

getch();

}
```

A4. WAP to sort an array of “n” elements in ascending order?

[HSEB-2065,2067] **10 marks**

```
#include<stdio.h>

#include<conio.h>

main( )
```

```
{
int i,j,n,num[100],temp;
printf("Enter the size of array not more than 100\n");
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter the %d number =",i+1);
scanf("%d",&num[i]);
}
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(num[i]>num[j])
{
temp=num[i];
num[i]=num[j];
num[j]=temp;
}

}

}

printf("Series in Ascending order\n");

for(i=0;i<n;i++)

{

printf("%d \t",num[i]);

}

getch( );                              }
```

A5. WAP to sort an array of “n” elements in descending order? [HSEB 2063,2068] 10 marks

```
#include<stdio.h>

#include<conio.h>

main( )

{

int i,j,n,num[100],temp;

printf("Enter the size of array not more than 100\n");
```

```
scanf("%d",&n);
for(i=0;i<n;i++)
{
printf("Enter the %d number =",i+1);
scanf("%d",&num[i]);
}
for(i=0;i<n;i++)
{
for(j=i+1;j<n;j++)
{
if(num[i]<num[j])
{
temp=num[i];
num[i]=num[j];
num[j]=temp;
}
}
}
```

```
printf("Series in Ascending order\n");

for(i=0;i<n;i++)

{

printf("%d \t",num[i]);

}

getch( );                    }
```

A6. WAP to input “n” numbers and find out the greatest and smallest number? [HSEB-2062]

```
#include<stdio.h>

#include<conio.h>

Void main()

{

int n, i, max, a[100], min;

printf("Enter the size of array not more than 100\n");

scanf("%d",&n);

for(i=0;i<n;i++)

{
```

```
printf("Enter %d number=",i+1);

scanf("%d",&a[i]);

}

max=a[0];

min=a[0];

for(i=0;i<n;i++)

{

if(a[i]>max)

max=a[i];

else if(a[i]< min )

min=a[i];

else

;

}

printf("Greatest number=%d\n",max);

printf("Smallest number=%d",min);

getch();

}
```

A7. WAP to input names of "n" numbers of students and sort them in alphabetical order? [HSEB-2062,2068] 10 marks

```
#include<stdio.h>

#include<conio.h>

#include<string.h>

main()

{

    char name[100][100],temp[100];

    int i,j,n;

    printf("Enter numbers of students");

    scanf("%d",&n);

    for(i=0;i<n;i++)

    {

    printf("Enter %d student name=",i+1);

    scanf("%s",name[i]);

    }
    for(i=0;i<n-1;i++)
    {
    for(j=i+1;j<n;j++)
    {
    if(strcmp(name[i],name[j])>0)
    {
    strcpy(temp,name[i]);
    strcpy(name[i],name[j]);
    strcpy(name[j],temp);
    }
    }
    }
    printf("Sorted Alphabetically\n");
    for(i=0;i<n;i++)
    {
    printf("%s\n",name[i]);
}
getch( );
```

```
}
```

A8. WAP which reads salary of 25 employees and count the number of employees who are getting salary between 30,000 to 40,000?

[HSEB- 2061, 2062, 2063, 2068, 2070] **10 marks**

```
#include<stdio.h>

#include<conio.h>

main()

{

int salary[25],i,c=0;

for(i=0;i<25;i++)

{

printf("Enter the salary of %d employee=\n ",i+1);

 scanf("%d",&salary[i]);

}
for(i=0;i<25;i++)
{
if(salary[i]>=30000 && salary[i]<=40000)
                {
                        c=c+1;
                }
        }
        printf("Total number of employee who are getting salary in the range of 30,000 to 40,000 are=%d",c);
getch();
}
```

WAP to read the elements of given two matrices of order 3x3 and to perform the matrix addition.

```
#include<stdio.h>
#include<conio.h>
int main(){
int m1[3][3],m2[3][3],m3[3][3],i,j;
```

```
printf("Enter the elements of the first matrix ");

for(i=0;i<3;i++)

{

        for(j=0;j<3;j++)

        {

                scanf("%d",&m1[i][j]);

        }

}

printf("Enter the elements of the second matrix ");

for(i=0;i<3;i++)

{

        for(j=0;j<3;j++)

        {

                scanf("%d",&m2[i][j]);

        }
```

```
}
        for(i=0;i<3;i++)
{
        for(j=0;j<3;j++)
        {
                m3[i][j]=m1[i][j]+m2[i][j];
        }
}
printf("The sum of two matrices  is: \n ");
for(i=0;i<3;i++)
{
        for(j=0;j<3;j++)
        {
                printf("%d\t",m3[i][j]);
        }
```

```
                printf("\n");
        }
getch();
return 0;
}
```

A9. WAP to add two matrices by supplying elements of matrices by the user?

[HSEB- 2065,2069] **10 marks**

```
#include<stdio.h>
#include<conio.h>
main()
{
        int m1[3][4],m2[3][4],m3[3][4],r,c;
        printf("Enter the elements of the first matrix ");
        for(r=0;r<3;r++)
        {
                for(c=0;c<4;c++)
                {
                        scanf("%d",&m1[r][c]);
        }
}
printf("Enter the elements of the second matrix ");
for(r=0;r<3;r++)
{
        for(c=0;c<4;c++)
        {
                scanf("%d",&m2[r][c]);
        }
}
        for(r=0;r<3;r++)
{
        for(c=0;c<4;c++)
        {
                m3[r][c]=m1[r][c]+m2[r][c];
        }
}
printf("The sum of two matrices  is ");
printf("\n");
```

```
        for(r=0;r<3;r++)
        {
                for(c=0;c<4;c++)
                {
                        printf("%d\t",m3[r][c]);
                }
                printf("\n");
        }
        getch();
}
```

A10. WAP to transpose a matrix?

```
#include<stdio.h>
#include<conio.h>
main()
{
int i,j,temp;
int a[3][3]={{1,2,3},{4,5,6},{7,8,9}};
```

```
for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

{

temp=a[i][j];

a[i][j]=a[j][i];

a[j][i]=temp;

}

}

printf("Transpose matrixes\n");

for(i=0;i<3;i++)

{

for(j=0;j<3;j++)

printf("%d\t",a[j][i]);

printf("\n");
}

getch();

}
```

Homework Questions

C-2 (Array)

2.1 What is an array? Differentiate between 1-D and 2-D array.

2.2 WAP to find the largest among ʺnʺ numbers?

2.3 WAP to store 10 numbers and print the largest among them?

2.4 WAP to sort an array of ʺnʺ numbers in ascending orders?

2.5 WAP to sort an array of ʺnʺ numbers in descending orders?

2.6 WAP to input ʺnʺ numbers and find out the greatest and smallest number?

2.7 WAP to input names of ʺnʺ numbers of students and sort them in alphabetical numbers?

2.8 WAP which reads salary of 25 employees and count the number of employees who are getting salary between 30,000 to 40,000?

2.9 WAP to add two matrices by supplying elements of matrices by the user?

2.10 WAP to transpose a matrix?

THE END

STRING

Definition: The collections of the characters are called string. It is always enclosed in double quotation marks. It contains characters, symbols, numbers etc. Example: ″Kathmandu″, ″12345″ etc.

String Handling Functions:

The c library supports many string handling functions that can be used to carry out many of the string manipulation. The header file ″string.h″ is used for string manipulation functions.

1. **Strlen(variable name)**: - This string function is used to find out the exact length of the string. The return of the strlen function is integer value.

Syntax:

n= strlen(str);

where str is the string and n is the length of string , returned by strlen function.

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
int l;
char str[100];
printf("Enter a string");
scanf("%s",str);
l=strlen(str);
printf("LENGTH=%d",l);
getch();
}
```

Output:

```
Enter a string RAMSIR

        LENGTH=6
```

2. **Strcat(destination,source):-** This string function is used to concatenate / merge two strings in first one.

Syntax:

strcat(str1,str2);

where str1,str2 are the two strings to be join together.

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char a[100],b[100];
printf("Enter string1,string 2");
scanf("%s%s",a,b);
strcat(a,b);
printf("After concatenating two string =%s",a);
```

Output:

```
Enter string1,string2

PK

SIR

  After concatenating two string =PKSIR
```

getch();
}	

3. Strcmp(string1,string2):- This is used to compare two string each other.
Syntax:

strcmp(str1,str2);

Where str1and str2 are two strings to be compared.

Example:

#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char a[100], b[100];
 	
printf("Enter string1,string 2");
 	
scanf("%s%s", a, b);
 	
if(strcmp(a,b)==0)
 	
printf("Both strings are same.");
 	
else
 	
printf("Both strings are not same.");
	Enter string1, string 2 KIST
	KIST
	Both strings are same.
	//
	Enter string1, string 2 KIST
	SIR
	Both strings are not same.

getch();	
}	

4. **strrev(variable name):-** This is used to reverse string.

Syntax:

strrev(str);

where str is the string to be reverse in order.

Example:

#include<stdio.h>	**Output:**
#include<conio.h>	Enter a string ABC
#include<string.h>	REVERSE ORDER=CBA
main()	
{	
char str[100];	
printf("Enter a string");	

```
scanf("%s",str);

printf("REVERSE ORDER=%s",strrev(str));

getch();

}
```

5. Strupr(string):- This is used to change case of characters.
Syntax:

strupr(str);

Where str is string to be converted into uppercase.

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char str[100];
printf("Enter a string");
scanf("%s",str);
printf("UPPER CASE=%s",strupr(str));
getch();
}
```

Output:

```
Enter a string abc

UPPER CASE=ABC
```

6. Strlwr(string):- This is used to change case of characters.
Syntax:

strlwr(str);

Where str is string to be converted into lowercase.

Example:

```
#include<stdio.h>
#include<conio.h>
#include<string.h>
main()
{
char str[100];
printf("Enter a string");
scanf("%s",str);
printf("lower case=%s",strlwr(str));
getch();
}
```

Output:

```
Enter a string CMP

lower case=cmp
```

7. **Strcpy (str1,str2):**- It is used to copy one string into other string.

Syntax:

strcpy(str1,str2);

Where str1,str2 are two strings and content of string2 is copied on a string str1.

Example:

#include<stdio.h>
 	
#include<conio.h>
 	
#include<string.h>
 	
main()
 	
{
 	
char str1[100], str2[100];
 	
printf("Enter a first string=");
 	
scanf("%s″, str1);
 	
strcpy(str2, str1);
 	
printf("After copying the string =%s",str2);
 	
getch();
 	
}
	Output:
	
	Enter a first string=ABC
	
	After copying the string =ABC

<u>Homework Questions</u>

<u>C-3 (String Function)</u>

3.1 Describe any five″string handling functions″ with examples.

3.2 What is string? Explain any four string handling functions with example.

3.3 WAP to read a line of text and to convert it into uppercase.

3.4 What do you mean by string manipulation? Explain about strcpy and strcat?

THE END

<u>Infinite loop</u>

The loops that do not end are called infinite loop. Generally, infinite loops are used in server code where the servers have to run without any interruption.

Example:

```
for ( ;    ; )
{
}
```