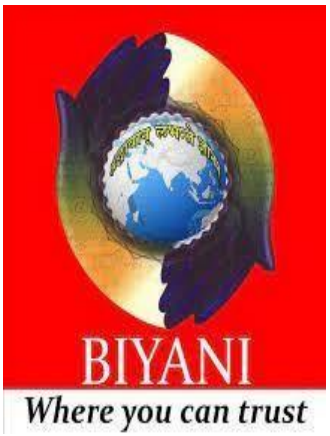
Biyani's Think Tank

Concept based notes

Programming in C

BCA I Sem.

Mr. Rahul Agarwal
AssociateProfessor
Dept. of IT
Biyani Girls College, Jaipur



Published by :
Think Tanks
Biyani Group of Colleges

Sector-3, Vidhyadhar Nagar,
Jaipur-302 023 (Rajasthan)
Ph : 0141-2338371, 2338591-95 Fax : 0141-2338007
E-mail : acad@biyanicolleges.org
Website :www.gurukpo.com; www.biyanicolleges.org

ISBN : 978-93-83462-22-3

Edition: 2023

Leaser Type Setted by :
Biyani College Printing Department

Preface

I am glad to present this book, especially designed to serve the needs ofthe students. The book has been written keeping in mind the general weakness in understanding the fundamental concepts of the topics. The book is self- explanatory and adopts the "Teach Yourself" style. It is based on question- answer pattern. The language of book is quite easy and understandable basedon scientific approach.

Any further improvement in the contents of the book by making corrections, omission and inclusion is keen to be achieved based on suggestions from the readers for which the author shall be obliged.

I acknowledge special thanks to Mr. Rajeev Biyani, *Chairman* & Dr. Sanjay Biyani, *Director* (*Acad.*) Biyani Group of Colleges, who are the backbones and main concept provider and also have been constant source of motivation throughout this Endeavour. They played an active role in coordinating the various stages of this Endeavour and spearheaded the publishing work.

I look forward to receiving valuable suggestions from professors of various educational institutions, other faculty members and students for improvement of the quality of the book. The reader may feel free to send in their comments and suggestions to the under mentioned address.

Author

Course Content for BCA, Semesters I and II

Semester: I

Corse Code: **BCA-51T-101**	Course Title: **Programming in C**
Course Credit : 04	Hours/Week: 04

Course Outcomes (COs):

After completing this course satisfactorily, a student will be able to:

- Confidently operate Desktop Computers to carry out computational tasks
- Understand working of Hardware and Software and the importance of operating systems
- Understand programming languages, number systems, peripheral devices, networking, multimedia and internet concepts
- Read, understand and trace the execution of programs written in C language
- Write the C code for a given problem
- Perform input and output operations using programs in C
- Write programs that perform operations on arrays Course Content

BCA-51T-101: Programming in C

UNIT- I

Basic concepts of Programming languages, Programming Domains, Language Evaluation criteria and language categories, Evolution of major programming languages. Describing syntax and semantics, formal methods of describing syntax, Pseudo code, Design of Algorithm & Flowchart

UNIT- II

Fundamentals of C: History and importance of C, basic structure and execution of C programs, constants, variables, and data types, Various type of declarations, operators types and expressions, evaluation of expressions, operator precedence and associability. Managing input and output operations, decision making and branching.

Iteration: while, do...while, for loop, nested loops, break & continue, goto statements.

UNIT- III

Array and String: One-dimensional array and their declaration and initialization, two dimensional arrays and their initializations, character arrays (One and Two dimensional), reading and writing strings, string - handling functions.

Functions: Need and elements for user –defined functions, definition of functions, return values and their types, function calls and declaration, recursion, parameter passing, passing arrays and strings to functions, the scope, visibility and life time of variables.

Registrar
(Academic)
University of Rajasthan
JAIPUR

UNIT-IV

Understanding Pointers: Accessing the address of a variable, declaration and initialization of pointer variables, accessing a variable through its pointer, pointers and arrays, pointers and function arguments, functions returning pointers.

Structures and Unions: Defining structure, declaring structure variable and accessing structure members, initialization of structure, operation on individual members, and array of structures, union, size of structure.

Recommended Books:

1. Balagurusamy E; Programming in ANSI C;Fifth Edn; Mc Graw Hill,2011.
2. Kanetkar Y.; LET US C; X Edition, BPB,2010.
3. Deitel HM & Deitel JP; C How to program; 5th Edn; Pearson Pub
4. Gottfried B; Programming with C: Schaum Qutlines; Mc Graw Hill Edition.

Corse Code: **BCA-51P-102**	Course Title: **Programming in C Lab**
Course Credit : 02	Hours/Week: 04

Content : **Recommended exercises**

Part A:

1. Program to read radius of a circle and to find area and circumference
2. Program to read three numbers and find the biggest of three
3. Program to demonstrate library functions in math.h
4. Program to check for prime
5. Program to generate n primes
6. Program to read a number, find the sum of the digits, reverse the number and check it for palindrome
7. Program to read numbers from keyboard continuously till the user presses 999 and to find the sum of only positive numbers
8. Program to read percentage of marks and to display appropriate message (Demonstration of else-if ladder)
9. Program to find the roots of quadratic equation (demonstration of switch Case statement)
10. Program to read marks scored by n students and find the average of marks (Demonstration of single dimensional array)
11. Program to remove Duplicate Element in a single dimensional Array
12. Program to perform addition and subtraction of Matrices

Part B:

1. Program to find the length of a string without using built in function
2. Program to demonstrate string functions.
3. Program to demonstrate pointers in C
4. Program to check a number for prime by defining isprime( ) function
5. Program to read, display and to find the trace of a square matrix
6. Program to read, display and add two m x n matrices using functions
7. Program to read, display and multiply two m x n matrices using functions
8. Program to read a string and to find the number of alphabets, digits, vowels, consonants, spaces and special characters.
9. Program to Reverse a String using Pointer

10. Program to Swap Two Numbers using Pointers
11. Program to demonstrate student structure to read & display records of n students.
12. Program to demonstrate the difference between structure & union.

Note: **Student has to execute a minimum of 10 programs in each part to complete the Lab course.**

Chapter 1

Introduction to Programming tit-bits

Q 1. How do we define a character set?

Ans Any alphabet ,digit or symbols to represent information is called Character .
The characters are grouped into following categories:

1 Letters
2 Digits
3 Special characters
4 White spaces

The following are the valid alphabets, numbers and special symbols permitted in C.
Digits: From 0 to 9
Letters: From a to z, A to Z.
Special characters : , . ? „ " / \
White space: Blank Spaces , Tab , New Line.

Q2 What are Identifiers?

Ans Identifiers" are the names that we supply for variables, types, functions, and labels in our program. Identifier names must differ in spelling and case from any keywords. We cannot use keywords (either C or Microsoft) as identifiers; they are reserved for special use. We create an identifier by specifying it in the declaration of a variable, type, or function.

Q3 Define Variables.

Ans A variable is a name given to the memory location for holding data. The name of the memory location i.e. the variable name, remain fixed during execution of the program but the data stored in that location may change from time to time.

Eg. Marks1 , Marks2, abc , a , ab_1 ,

Rules for writing variable names

1. The first character of variable name must be an alphabetic.
2. Blank spaces are not allowed in a variable name.
3. Special characters such as arithmetic operators, #,^ can not be used in a variable.
4. Reserved words(Keywords) cannot be used as variable names.

5. The maximum length of a variable name depends upon the compiler (8).
6. A variable name declared for one data type cannot be used to declare another data type.

Q4 What do you mean by constants?

Ans Constant is fixed value which can not be changed by the program during the execution.

Eg. A=5 in this 5 is constant.

Q 5. How many types of constants ?

A2. There are mainly three types of constants namely: integer, real and character constants.

1. Integer Constants:
   - (i) Decimal Integer Constant:
     0 to 9
     E.g: 49, 58, -62, ... (40000 cannot come bcoz it is > 32767)
   - (ii) Octal Integer Constant:
     0 to 7
     Add "0" before the value.
     Eg.: 045, 056, 067
   - (iii) Hexadecimal Integer:0 to 9 and A to F Add 0x before the value
     E.g: 0x42, 0x56, 0x67
2. Real Constants:
   The real or floating point constants are in two forms namely fractional form and the exponential form.
   A real constant in fractional form must have a digit with a decimal part.Ex 456.78
   In exponential form, the real constant is represented as two parts. The part lying before the „e" is the „mantissa", and the one following „e" is the „exponent".
   Ex: +3.2e-4, 4.1e8, -0.2e+4, -3.2e-4
3. Character Constants
   A character constant is an alphabet, a single digit or a single special symbol enclosed within inverted commas. Ex: "B", "I", "#"

Q 6. What are key words?

A.ns They are the reserved words that cannot be used for naming a variable. They perform fix tasks.

Eg. int , char , for , if .

Q7 Explain Instructions.

Ans C instruction are of basically three types :

1 Type declaration instruction
2 Arithmetic Instruction
3 Control Instruction

Type Declaration Instructions

This instruction is used to declare the type of variables being used in the program. Any variable used in the program must be declared before using it in any statement. The type declaration statements is written at the beginning of the main() function.

The main purpose of type declaration instruction is to declare the type of variable C program.

For Example :

```
int num;
char c;    // Type Declaration
float f;
main()
{
Some Statements
}
```

The Arithmetic Instruction

A C arithmetic instruction consists of a variable name on the left hand side of = and constants appearing on the right hand side of = are connected by arithmetic operators like +, -, * and /.

A C arithmetic statement could be of 3 types :

(a) Integer mode arithmetic statement : This is an arithmetic statement which all operands are either integer or integer constants.

For Example :

```
int i, j, l, m;
i=i+1;
m=i* j +l;
```

(b) Real Mode Arithmetic Statement : These are arithmetic statement in which all operands are either real constant or real variable.

For Example :

```
float si, roi, p, q ;
```

si = roi*p*q/100.0;

c) Mixed mode arithmetic statements : this is an arithmetic statement in which some of the operands are integer and some of the operands are real.

For Example :

int a, b, c, num ;

avg = (a + b+ c + num)/4;

Control instruction:

To control the sequence of execution of various statements in a C program.

Q8. What is Expression?

Ans Expression is any valid combination of operators, constants, functions and variables.

Statements like a = b + 3, ++z and 300 > (8 * k) are all expressions.

Q9 Discuss various operator in C and C++.

Ans An operator is a symbol that operates on a certain data type and produces the output as the result of the operation.

Eg. expression *4 + 5 is equal to 9*. Here 4 and 5 are called operands and + is called operator.

Category of operators

Unary Operators:-A unary operator is an operator, which operates on one operand.

Binary:-A binary operator is an operator, which operates on two operands

Ternary:-A ternary operator is an operator, which operates on three operands.

C contains the following operator groups

1 Arithmetic Operator

The arithmetic operator is a binary operator, which requires two operands to perform its operation of arithmetic. Following are the arithmetic operators that are available.

Operator	Description	Eg.
+	Addition	a+b
-	Subtraction	a-b
/	Division	a/b
*	Multiplication	a*b
%	Modulo or remainder	a%b

2 Relational Operators

Relational operators compare between two operands and return in terms of true or false i.e. 1 or 0. In C and many other languages a true value is denoted by the integer 1 and a false value is denoted by the integer 0. Relational operators are used in conjunction with logical operators and conditional & looping statements.

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

3 Logical Operators

A logical operator is used to compare or evaluate logical and relational expressions. There are three logical operators available in the C language.

&& Logical AND
||Logical OR
!Logical NOT

4 Assignment operator

An assignment operator (=) is used to assign a constant or a value of one variable to another.

Example:

```
a = 5;
b = a;
rate = 10.5
net = (a/b) * 100;
```

* There is always difference between the equality operator (==) and the assignment operator (=).

5 Conditional or Ternary Operator

A conditional operator checks for an expression, which returns either a true or a false value. If the condition evaluated is true, it returns the value of the true section of the operator, otherwise it returns the value of the false section of the operator.

Its general structure is as follows:

Expression1 ? expression 2 (True Section): expression3 (False Section) Example:

```
a=3,b=5,c;
```

c = (a>b) ? a+b : b-a;

The variable c will have the value 2, because when the expression (a>b) is checked, it is evaluated as false. Now because the evaluation is false, the expression b-a is executed and the result is returned to c using the assignment operator.

6 Bitwise Operators:

These are used to perfom bitwise operations such as testing the bits , shifting the bits to left or right , one"s compliment of bits. This operator can be apply on only int and char data type.

& AND
| Inclusive OR
^ Exclusive OR
<< Shift Left >>
Shift Right
~ One's compliment
~A = 1100 0011

7 Increment and Decrement Operators

These operators are unary operators .

The increment and decrement operators are very useful in C language. They are extensively used in for and while loops. The syntax of these operators is given below.

++
--

8 Comma operator (,):-

The comma operator (,) is used to separate two or more expressions that are included where only one expression is expected. When the set of expressions has to be evaluated for a value, only the rightmost expression is considered.

Q10 What are the ways to comment statement in C?

Ans Comments are non executable statement

Most of C/C++ will support two types of comments:

// Comment text goes here (in line)

/* Comment goes here */ (**block**)

Q11 Input and output statements

Ans

Input :

In any programming language input means to feed some data into program. This can be given in the form of file or from command line. C programming language provides a set of built-in functions to read given input and feed it to the program as per requirement.

printf() function

This is one of the most frequently used functions in C for output

Output :

In any programming language output means to display some data on screen, printer or in any file. C programming language provides a set of built-in functions to output required data.

scanf() function

This is the function which can be used to to read an input from the command line.

Chapter 2

The Decision , Loop , Case Control Structure

Q.1 Explain control structures available in C and C++.

A C provides two styles of flow control: Branching
Looping

Branching or Decision :

Branching or Decision is so called because the program chooses to follow one branch or another.

if statement

This is the most simple form of the branching statements.It takes an expression in parenthesis and an statement or block of statements. if the expression is true then the statement or block of statements gets executed otherwise these statements are skipped.

Syntex:- if with single statement

```
if (expression)
  statement;
```

Syntex:- if with block statement

```
if (expression)
  {
    Block of statements;
  }
```

or

Syntex:- if with else statement

```
if (expression)
  {
    Block of statements;
  }
else
```

```
 {
   Block of statements;
 }
```

Or

Syntex:- if with else if statement

```
if (expression)
 {
   Block of statements;
 }
else if(expression)
 {
   Block of statements;
 }
else
 {
   Block of statements;
 }
```

Eg.

```
#include <stdio.h>

main()
{
 int cows = 6;

 if (cows > 1)
   printf("We have cows\n");

 if (cows > 10)
   printf("loads of them!\n");
 else
   printf("Executing else part...!\n");

 if (cows == 5 )
 {
   printf("We have 5 cows\n");
 }
 else if( (cows == 6 )
```

```
  {
    printf("We have 6 cows\n");
  }
}
```

Output

```
We have cows
Executing else part...!
We have 6 cows
```

? : Operator

The ? : operator is just like an if ... else statement except that because it is an operator you can use it within expressions.

? : is a ternary operator in that it takes three values, this is the only ternary operator C has.

? : takes the following form:

if condition is true ? then X return value : otherwise Y

value; switch statement:

The switch statement is much like a nested if .. else statement. Its mostly a matter of preference which you use, switch statement can be slightly more efficient and easier to read.

```
switch( expression )
   {
        case expression1:
        statements1;
        case expression2:
        statements2;
        case c-expression3:
        statements3;
        default : statements4;
   }
```

Use of break

Use If a condition is met in switch case then execution continues on into the next case clause also if it is not explicitly specified that the execution should exit the switch statement. This is achieved by using **break keyword.**

Looping

Loops provide a way to repeat commands and control how many times they are repeated. C provides a number of looping way.

while loop

The most basic loop in C is the while loop. Like an If statement, if the test condition is true, the statements get executed. The difference is that after the statements have been executed, the test condition is checked again. If it is still true the statements get executed again.This cycle repeats until the test condition evaluates to false.

syntax

```
while ( expression )
{
  Single statement
  or
  Block of statements;
}
```

for loop

for loop is similar to while, it's just written differently. for statements are often used to proccess lists such a range of numbers:

syntax:

```
for( expression1; expression2; expression3)
{
  Single statement
  or
  Block of statements;
}
```

In the above syntax:

expression1 - Initialisese variables.

expression2 - Condtional expression, as long as this condition is true, loop will keep executing.

expression3 - expression3 is the modifier which may be simple increment of a variable.

do...while loop

do ... while is just like a while loop except that the test condition is checked at the end of the loop rather than the start. This has the effect that the content of the loop are always executed at least once.

syntax

```
do
{
  Single statement
  or
  Block of statements;
```

```
}while(expression);
```

break and continue statements

C provides two commands to control the loop:
break -- exit form loop or switch.
continue -- skip 1 iteration of loop.

```
#include
main()
{
   int i;
   int j = 10;

   for( i = 0; i <= j; i ++ )
   {
     if( i == 5 )
     {
       continue;
     }
     printf("Hello %d\n", i );
   }
}
```

```
Hello 0
Hello 1
Hello 2
Hello 3
Hello 4
Hello 6
Hello 7
Hello 8
Hello 9
Hello 10
```

The goto statement (unconditional branching)

goto allows to make an absolute jump to another point in the program. We should use this feature with caution since its execution causes an unconditional jump ignoring any type of nesting limitations.

The destination point is identified by a label, which is then used as an argument for the goto statement. A label is made of a valid identifier followed by a colon (:).

goto loop example

```
#include <stdio.h>
int main ()
{
int n=10;
loop:
printf("%d", n);
n--;
if (n>0)
goto loop;
printf( "FIRE");
}
```

10, 9, 8, 7, 6, 5, 4, 3, 2, 1, FIRE!

exit function

exit is a function defined in the cstdlib library.

The purpose of exit is to terminate the current program with a specific exit code. Its prototype is:

```
exit()
```

Chapter 3
Functions

Q1 What is a Function?

Ans The function is a self contained block of statements which performs a task of a same kind. C program does not execute the functions directly. It is required to invoke or call that functions. When a function is called in a program then program control goes to the function body. Then, it executes the statements. We call function whenever we want to process that functions statements i.e. more than 1 times. Any c program contains at least one function. Function is used to avoids rewriting the same code over and over.

The following is its format:

type name (parameter1, parameter2, ...) { statements
} where:

- type is the data type specifier of the data returned by the function.
- **name** is the identifier by which it will be possible to call the function.
- parameters (as many as needed): Each parameter consists of a data type specifier followed by an identifier.
- statements is the function's body. It is a block of statements surrounded by braces { }.

Eg.

```
void add()
{
        int a, b, c;
        clrscr();
        printf("\n Enter Any 2 Numbers : ");
        scanf("%d %d",&a,&b);
        c = a + b;
        printf("\n Addition is : %d",c);
}
void main()
{
        void add();
        add();
```

```
        getch();
}
```

Q2 What are the properties of functions in C? Ans

function in a C program has some properties.

1 Every function has a unique name. This name is used to call function from "main()" function. A function can be called from within another function.

2 A function is independent and it can perform its task without intervention from or interfering with other parts of the program.

3 A function performs a specific task. A task is a distinct job that our program must perform as a part of its overall operation, such as adding two or more integer.

4 A function returns a value to the calling program. This is optional and depends upon the task your function is going to accomplish..

Q3 What are the types of functions ?

Ans There are 2(two) types of functions as:

1. Built in Functions
2. User Defined Functions

1. Built in Functions :

These functions are also called as 'library functions'. These functions are provided by system. These functions are stored in library files. e.g.

scanf()
printf()
strcpy

2 User Defined Functions :

The functions which are created by user for program are known as 'User defined functions'.

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

void add()
{
        int a, b, c;
        clrscr();
        printf("\n Enter Any 2 Numbers :
        "); scanf("%d %d",&a,&b);
        c = a + b;
```

```
        printf("\n Addition is : %d",c);
}
void  main()
{
        void add();
        add();
        getch();
}
```

Q4 Write Parameter passing mechanisms in C?

Ans There are two ways to pass parameters to a function:

Pass by Value: mechanism is used when you don't want to change the value of passed paramters. When parameters are passed by value then functions in C create copies of the passed in variables and do required processing on these copied variables.

```
int main()
{
  int a = 10;
  int b = 20;

  printf("Before: Value of a = %d and value of b = %d\n", a, b
  ); swap( a, b );
  printf("After: Value of a = %d and value of b = %d\n", a, b );
}

void swap( int p1, int p2 )
{
   int t;

   t = p2;
   p2 = p1;
   p1 = t;
  printf("Value of a (p1) = %d and value of b(p2) = %d\n", p1, p2 );
}
```

Before: Value of a = 10 and value of b = 20
Value of a (p1) = 20 and value of b(p2) = 10
After: Value of a = 10 and value of b = 20

Pass by Reference : This mechanism is used when you want a function to do the changes in passed parameters and reflect those changes back to the calling function. In this case only addresses of the variables are passed to a function so that function can work directly over the addresses.

```
void swap( int *p1, int *p2 );

int main()
{
  int a = 10;
  int b = 20;

  printf("Before: Value of a = %d and value of b = %d\n", a, b
  ); swap( &a, &b );
  printf("After: Value of a = %d and value of b = %d\n", a, b );
}

void swap( int *p1, int *p2 )
{
   int t;

   t = *p2;
   *p2 = *p1;
   *p1 = t;
  printf("Value of a (p1) = %d and value of b(p2) = %d\n", *p1, *p2 );
}
```

before: Value of a = 10 and value of b = 20
Value of a (p1) = 20 and value of b(p2) = 10
After: Value of a = 20 and value of b = 10

Q5 Scope and lifetime of variables.

Ans

Storage Class	Storage	Default initial value	Scope	Life
Automatic	Memory	Garbage	Local to the block in which variable is defined	Till the control remains within the block in which the variable is

				defined
Register	CPU registers	Garbage	Local to the block in which variable is defined	Till the control remains within the block in which the variable is defined
Static	Memory	Zero	Local to the block in which variable is defined	Value of variable persists between different function calls.
External	Memory	Zero	Global	As long as program execution doesn"t come to end.

Chapter 4

Array

Q1. Define arrays.

Ans. A collection of variables which are all of the same type. It is a data structure, which provides the facility to store a collection of data of same type under single variable name. Just like the ordinary variable, the array should also be declared properly. The declaration of array includes the type of array that is the type of value we are going to store in it, the array name and maximum number of elements.

Examples:

short val[200]; //declaration

val[12] = 5; //assignment

Q2. How is Array declared?

Ans Declaration & Data Types

Arrays have the same data types as variables, i.e., short, long, float etc.They are similar to variables: they can either be declared global or local.They are declared by the given syntax:

Datatype array_name[dimensions]={element1,element2,....,element}

Q3. Write a program for one dimensional array and two dimensional array.

Ans The declaration form of one-dimensional array is

Data_type array_name [size];

The following declares an array called „numbers" to hold 5 integers and sets the first and last elements. C arrays are always indexed from 0. So the first integer in „numbers" array is numbers[0] and the last is numbers[4].

```
int numbers [5];
numbers [0] = 1;      // set first element
numbers [4] = 5;      // set last element
```

This array contains 5 elements. Any one of these elements may be referred to by giving the name of the array followed by the position number of the particular element in square brackets ([]). The first element in every array is the zeroth element. Thus, the first element of array „numbers" is referred to as numbers[0], the second element of array „numbers" is referred to as numbers[1], the fifth element of array „numbers" is referred to as numbers[4

], and, in general, the n-th element of array „numbers‟ is referred to as numbers[n - 1].

Example:

```
#include   <stdio.h>
#include   <conio.h>
int main( )
    {
     char name[7]; /* define a string of characters */
     name[0] = 'A';
     name[1] = 's';
     name[2] = 'h';
     name[3] = 'r';
     name[4] = 'a';
     name[5] = 'f';
     name[6] = '\0'; /* Null character - end of text
     */ name[7] = „X‟;
     clrscr();
     printf("My name is %s\n",name);
     printf("First letter is %c\n",name[0]);
     printf("Fifth letter is %c\n",name[4]);
     printf("Sixth letter is %c\n",name[5]);
     printf("Seventh letter is %c\n",name[6]);
     printf("Eight letter is %c\n",name[7]);
     getch();
     return 0;
    }
```

Output

```
My name is Ashraf
First letter is A
Fifth letter is a
Sixth letter is f
Seventh letter is Null
Eight letter is X
```

Two dimensional array

Two-dimensional array are those type of array, which has finite number of rows and finite number of columns. The declaration form of 2-dimensional array is

Data_type Array_name [row size][column size];

Example:

```
 #include<stidio.h>
#include<conio.h>
int main()
{
int matrix[3][3],I,j,r,c;
clrscr();
printf("Enter the order of matrix\n");
scanf("%d%d",&r,&c);
printf("Enter the elements of 3x3 matrix\n",r,c);
for(i=0;i<r;i++)
        for(j=0;j<c;j++)
                scanf("%d",&matrix[i][j]);
printf("Given matrix:\n");
for(i=0;i<r;i++)
        for(j=0;j<c;j++)
        printf("%d\t",matrix[i][j]);
printf("\n");
}
getch();
return 0;
}
```

Output

1 2 3
2 3 4
5 6 7

Q.4 Explain String.

Ans A group of characters stored in a character array. A string in C is a sequence of zero or more characters followed by a NULL '\0' character:
String constants have double quote marks around them,. Alternatively, we assign a string constant to a char array - either with no size specified, or you can specify a size, but don't forget to leave a space for the null character!. **Eg.**

```
char string_2[] = "Hello";
        char string_3[6] = "Hello";.
```

Chapter 5

Structure, Arrays and Union

Q1. Define a Structure with suitable program.

Ans . A structure is a user defined data type. We know that arrays can be used to represent a group of data items that belong to the same type, such as int or float. However we cannot use an array if we want to represent a collection of data items of different types using a single name. A structure is a convenient tool for handling a group of logically related data items.

The syntax of structure declaration is

```
struct structure_name
    {
    type element 1;
    type element 2;
    ……..
    type element n;
    };
```

In structure declaration the keyword struct appears first, this followed by structure name. The member of structure should be enclosed between a pair of braces and it defines one by one each ending with a semicolon. It can also be array of structure. There is an enclosing brace at the end of declaration and it end with a semicolon.

We can declare structure variables as follows

struct structure_name var1,var2,…..,var n;

For Example:

To store the names, roll number and total mark of a student you can declare 3 variables. To store this data for more than one student 3 separate arrays may be declared. Another choice is to make a structure. No memory is allocated when a structure is declared. It just defines the "form" of the structure. When a variable is made then memory is allocated. This is

equivalent to saying that there's no memory for “int” , but when we declare an integer that is. int var; only then memory is allocated. The structure for the above-mentioned case will look like

```
struct student
        {
        int rollno;
        char name[25];
        float totalmark;
        };
```

We can now declare structure variables stud1, stud2 as follows struct student stud1,stud2;

Thus, the stud1 and stud2 are structure variables of type student. The above structure can hold information of 2 students.

It is possible to combine the declaration of structure combination with that of the structure variables, as shown below.

```
struct structure_name
        {
        type element 1;
        type element 2;
         ……..
        type element n;
        }var1,var2,...,varn;
```

The following single declaration is equivalent to the two declaration presented in the previous example. struct student

```
        {
        int rollno;
        char name[25];
        float totalmark;
        } stud1, stud2;
```

The different variable types stored in a structure are called its members. The structure member can be accessed by using a dot (.) operator, so the dot operator is known as structure member operator.

Example:

In the above example stud1 is a structure variable of type student. To access the member name, we would write

stud1.name

Similarly, stud1"s rollno and stud1"s totalmark can be accessed by writing

stud1.rollno And

stud1.totalmark

Q 2. How is Initialization of sructure members carried out?

Ans Structure members can be initialized at declaration. This much the same manner as the element of an array; the initial value must appear in the order in which they will be assigned to their corresponding structure members, enclosed in braces and separated by commas .The general form is

struct stucture_name var={val1,val2,val3.....};

Example:

```
#include <stdio.h>
#include<conio.h>
int main()
  {
  struct student
        {
        char *name;
        int rollno;
        float totalmark;
       };
  struct student stud1={"Ashraf",1,98};
  struct student stud3= {"Rahul",3,97};
  struct student stud2={"Vineeth",2,99};
  clrscr();
  printf("STUDENTS DETAILS:\nRoll
```

```
                number:%d\n\nName:%s\n\nTotal mark:%.2f\n",
        stud1.rollno,stud1.name,stud1.totalmark);

    printf("\nRoll number:%d\n\nName:%s\n\nTotel

        mark:%.2f\n",stud2.rollno,stud2.name,stud2.totalmark);

    printf("\nRoll number:%d\n\nName:%s\n\nTotel

        mark:%.2f\n",stud3.rollno,stud3.name,stud3.totalmark);

  getch();
  return 0;
  }
```

Output

Roll Number 1

Name Ashraf

Total Marks 98

Roll Number 3

Name Rahul,

Total Marks 97

Roll Number 2

Name Vineet

Total Marks 9

Q 3. Is it possible to creat Array of structures:?

Ans It is possible to store a structure has an array element. i.e., an array in which each element is a structure. Just as arrays of any basic type of variable are allowed, so are arrays of a given type of structure. Although a structure

contains many different types, the compiler never gets to know this information because it is hidden away inside a sealed structure capsule, so it can believe that all the elements in the array have the same type, even though that type is itself made up of lots of different types.

The declaration statement is given below.

```
struct struct_name
    {
    type element 1;
    type element 2;
    ……..
    type element n;
    }array name[size];
```

Example:

```
struct student
    {
    int rollno;
    char name[25];
    float totalmark;
    } stud[100];
```

In this declaration stud is a 100-element array of structures. Hence, each element of stud is a separate structure of type student. An array of structure can be assigned initial values just as any other array. So the above structure can hold information of 100 students.

Program:

```
#include  <stdio.h>
#include <conio.h>
int main()
    {
    struct student
          {
          int rollno;
          char name[25];
          int totalmark;
          }stud[100];
```

```
int n,i;
clrscr();
printf("Enter total number of students\n\n");
scanf("%d",&n);
for(i=0;i<n;i++)
        {
        printf("Enter details of %d-th student\n",i+1);
        printf("Name:\n");
        scanf("%s",&stud[i].name);
        printf("Roll number:\n");
        scanf("%d",&stud[i].rollno);
        printf("Total mark:\n");
        scanf("%d",&stud[i].totalmark);
        }
printf("STUDENTS DETAILS:\n");
for(i=0;i<n;i++)
        {
        printf("\nRoll number:%d\n",stud[i].rollno);
       printf("Name:%s\n",stud[i].name);
       printf("Totel mark:%d\n",stud[i].totalmark); }

getch();
return 0;
}
```

Output will be the dynamic data entered by the user of all the students.

Q 4. Explain the functionality of union.

Ans Union is a data type with two or more member similar to structure but in this case all the members share a common memory location. The size of the union corresponds to the length of the largest member. Since the member share a common location they have the same starting address.

The real purpose of unions is to prevent memory fragmentation by arranging for a standard size for data in the memory. By having a standard data size we can guarantee that any hole left when dynamically allocated

memory is freed will always be reusable by another instance of the same type of union. This is a natural strategy in system programming where many instances of different kinds of variables with a related purpose and stored dynamically.

A union is declared in the same way as a structure.The syntax of union declaration is

```
union union_name
      {
       type element 1;
       type element 2;
       ……..
       type element n;
      };
```

This declares a type template. Variables are then declared as:

union union_name x,y,z;

For example, the following code declares a union data type called Student and a union variable called stud: union student

```
        {
        int rollno;
        float totalmark;
        };
union student stud;
```

It is possible to combine the declaration of union combination with that of the union variables, as shown below. union union_name

```
        {
        type element 1;
        type element 2;
        ……..
        type element n;
        }var1,var2,…,varn;
```

The following single declaration is equivalent to the two declaration presented in the previous example. union student

```
{
 int rollno;
 float totalmark;
}x,y,z;
```

Q 5. Compare structure and Union

Ans Union allocates the memory equal to the maximum memory requried by the member of the union but structure allocates the memory equal to sum of the memory allocated to its each individual members.

In Union, one block is used by all the member of union but in case of structure, each member have their own memory space.

Example:

Structure:

```
#include  <stdio.h>

#include <conio.h>

int main()

{

        struct testing

{

int a;

char b;

float c;

}var;

clrscr();

printf("Size of var=%d\n",sizeof(var));

printf("Size of a=%d\n",sizeof(var.a));
```

```
printf("Size of b=%d\n",sizeof(var.b));

printf("Size of c=%d\n",sizeof(var.c));

var.a=10;

var.b="w";

var.c=3.1422;

printf("value of a=%d\n",var.a);

printf("value of b=%c\n",var.b);

printf("value of c=%f\n",var.c);

getch();

return 0;

}
```

Union:

```
#include  <stdio.h>

#include <conio.h>

int main()

{

union testing

{

int a;

char b;
```

```
float c;                                                    ( )

}var;

clrscr();

printf("Size of
var=%d\n",sizeof(var));

printf("Size of
a=%d\n",sizeof(var.a));

printf("Size of
b=%d\n",sizeof(var.b));

printf("Size of
c=%d\n",sizeof(var.c));

var.a=10;

var.b="w";

var.c=3.1422;

printf("value of a=%d\n",var.a);

printf("value of b=%c\n",var.b);

printf("value of c=%f\n",var.c);

getch();

return 0;

}
```

Output of both these programs will be same. The difference lies in the storage.

Chapter 6
Pointers

Q 1. Explain Pointer.

Ans . A pointer refers to a memory location that contains an address.

Pointers: Operators (1)

Address Operator: &

Note: it looks identical to the bitewise AND operator but it is used in a completely different way!Returns the address of a variable

Example: prt_v = & x;

Pointers: Operators (2)

Indirection Operator: *

Note: it looks identical to the multiplication operator but it is used in a completely different way!

Retrieves a value from the memory location the pointer points to.

Example: *ptr_v = 77;

Pointer Declaration

A pointer must be declared and the variable type it points to must be specified:

short *aptr; //pointer declaration

double *bptr;

Assigning an Address to a Pointer

short x = 33;

short *aptr; //pointer declaration

aptr = & x;

Each time we declare an array, we also declare implicitly a pointer to the "zeroth" element!

Q 2. Write a program to pass arguments by value and by refrence.

Ans . #include <ansi_c.h>

```
float KE_by_val( float a, float b);
float KE_by_ref( float *a, float *b);
main()
{
float q = 3, v = 5;
float *qptr, *vptr;
qptr = &q;
vptr = &v;
printf("%f\n", KE_by_val(q,v) );
printf("%f\n", KE_by_ref(&q,&v) );
printf("%f\n", KE_by_ref(qptr,vptr) );
}
float KE_by_val( float a, float b)
{
return( a * b);
}
float KE_by_ref( float *a, float *b)
{
return( (*a) * (*b));
}
```

Output

```
15.00
15.00
15.00
( )
```

This question paper contains 3 printed pages] Roll No. ——

Sl.No. **134**

B.C.A. (Part - I)

B.C.A. (Part - I) EXAMINATION, 2017
(Faculty of Science)
(Three - Year Scheme of 10 +2 + 3 Pattern)
Paper - 134
PRINCIPLES OF PROGRAMMING LANGUAGE (THROUGH 'C')

Time : Three Hours] *[Maximum Marks : 100*

Answer of all the questions (short answer as well as descriptive) are to be given in the main answer -book only Answers of short answer type questions must be given in sequential order Similarly all the parts of one question of descriptive part should be answered at one place in the answer-book One complete question should not be answered at different places in the answer-book. Write your roll numbers on question paper before start writing answers of questions.

PART - I:	*(Very Short Answer) consists of 10 questions of 2 marks each. Maximum limit for each question is up to 40 words.*
PART - II:	*(Short answer) consists of 5 questions of 4 marks each. Maximum limit for each question is up to 80 words.*
PART - III:	*(Long answer) consists of 5 questions of 12 marks each with internal choice.*

PART - I

Attempt all Questions

Each questions carries 2 marks **[10 × 2 = 20]**

1. a) What is algorithm?
   b) Give flow chart symbols for I/O, processing terminal and flow lines.
   c) How do we create constants in 'C'? Give syntax.
   d) What are local variables?
   e) Discuss purpose and syntax of goto statement.
   f) How do we read and write strings in 'C' Explain.
   g) What are formal parameters?

h) Define pointers.

i) How do we create structures in 'C'? Explain.

j) Differentiate between fprintf( ) and printf( ).

PART - II

Attempt all questions

Each questions carries 4 marks [5 × 4 = 20]

2. a) Draw a flow chart to find out sum and average of any 3 nos.

b) Discuss any 2 (two) data types of 'C' with suitable examples.

c) Differentiate between break and continue statements with the help of appropriate example(s).

d) Differentiate between call by value and call by reference.

e) Discuss the purpose of following functions:

i) putch ( )

ii) puts ( )

iii) putchar ( )

iv) scanf ( )

PART - III

3. Discuss machine level, Assembly and high level languages is detail. [12]

OR

Write pseudocodes to find out:

a) factorial of a given no.

b) sum of 1st 10 natural no's. [6 + 6 = 12]

4. Discuss the various operators of 'C'. [12]

OR

Write a program in 'C' to findout grade of a student based on the following criterias:

a) Percentage is < = 40; grade is 'D'.

b) Percentage is > = 40 but < 50; grade is 'C'.

c) Percentage is > = 50 but < 60; grade is 'B'.

d) Percentage is > = 60 but < 75 ; grade is 'A'.

e) Percentage is > = 75 grade is 'A+'.

5. Explain the following functions:
   a) streat ( )
   b) stremp ( )
   c) strempi ( )
   d) strlen ( )
   e) strstr ( )
   f) strchr( )

[6 × 2 =12]

OR

Discuss single Dimensional and double Dimensional arrays of 'C' in breif. [12]

6. What is recursion? Why do we use recursion? Explain Also write a code to print fibonacci series with recursive function. https://www.uoronline.com [12]

OR

Write a 'C' program that uses of function to search a no with in an array. [12]

7. Explain the following:
   a) File modes.
   b) Steps of file handling in 'C'.
   c) Stream I/O model.

[3 × 4 = 12]

OR

Create a structure containing five members : rollno, name - of- student, marks1, marks2 & marks3. Write a program to access those members using structure variable or pointer. [12]

B.C.A. (PART-I) EXAMINATION - 2018 Pri. of Pro. Lan. (Through C)
(Faculty of Science)
(Three year Scheme of 10+2+3 Pattern)
Paper - 134

PRINCIPLES OF PROGRAMMING LANGUAGE (THROUGH 'C')

Time Allowed : Three Hours
Maximum Marks - 100

PART I : (Very short answer) consists of 10 questions of 2 marks each. Maximum limit for each question is up to 40 words.

PART II : (Short answer) consists of 5 questions of 4 marks each. Maximum limit for each question is up to 80 words.

PART III : (Long answer) consists of 5 questions of 12 marks each with internal choice.

PART - I

1. Attempt all questions. Each question carries 2 marks. [10 x 2 = 20]
   (a) What is Pseudo Code?
   (b) Explain Programming Domains.
   (c) What is operator precedence?
   (d) What are the different methods to declare a cohstant in 'c'? Give example.
   (e) What is the difference between a do-while loop and a while loop?
   (f) What is the difference between a structure and a Union?
   (g) What is a NULL pointer? Give example.
   (h) What is enumerated data type? Give an example.
   (i) What is the difference between actual and formal parameters?
   (j) How do we calculate the size of a union in C?

PART - II

2. Attempt all questions. Each question carries 4 marks. [5 x 4 =20]
   (a) Draw a flowchart to calculate factorial of a given integer number.
   (b) Describe the skeleton of a 'C' program.
   (c) Write a program to read N values in an array and then find highest value.
   (d) Explain scope, visibility and lifetime of a variable in context to functions.

(c) Discuss the purpose of the following library functions :

(i) fseek()

(ii) rewind()

(iii) feof()

(iv) ftell

PART - III

3. (a) Write pseudo code to find the sum of first 10 even numbers. [12]

OR

(b) Write an algorithm to check whether a number entered by user is prime or not. [12]

4. (a) Explain different data types available in 'C'. [12]

OR

(b) Write a program to input basic salary of an employee and calculate its gross salary according to the following :

(i) Basic Salary <=10000 : HRA=20%, DA=80%

(ii) Basic Salary <=20000 : HRA=25%, DA=90%

(iii) Basic Salary >=20000 : HRA=30%, DA=95% [12]

5. (a) Write a 'C' program to find the length of a string without using built-in functions. [12]

OR

(b) Discuss the following :

(i) Declaration of one dimensional and two dimensional arrays.

(ii) Initialization of one dimensional and two dimensional arrays.

(iii) Accessing of elements from one dimensional and two dimensional arrays.

(iv) Why array name is called a constant pointer? [3 x 4 = 12]

6. (a) What do you mean by Recursion? Write a recursive program in 'C' to print all the elements of a mean array. [2 + 10 = 12]

OR

(b) What are Pointers? How a function can be called by using a pointer to it? Explain with an example. [2 + 10 = 12]

7. (a) Write a program in C to copy content of a file to another file. [12]

OR

(b) Explain the following :

(i) Declaring a structure

(ii) Accessing members of a structure using pointer

(iii) Self-referential structure

(iv) Difference between a structure and a union [3 x 4 = 12]

This question paper contains 2 printed pages.

Roll No. .....

B.C.A. (Pt. -I)

Pri. of Pro. Lan. (Through C)

134

B.C.A (Part-I) EXAMINATION, 2019

(Faculty of Science)

(Three Year Scheme of 10+2+3 Pattern)

PRINCIPLES OF PROGRAMMING LANGUAGE (THROUGH 'C') - 134

Time Allowed : Three Hours

Maximum Marks : 100

Answer all the questions (short answer as well as descriptive) are to be given in the main answer-book only. Answers of short answer type questions must be given in sequential order. Similarly, all the parts of one question of descriptive part should be answered at one place in the answer-book. One complete question should not be answered at different places in the answer-book.

Write your roll numbers on question paper before start writing answers of questions.

PART - I : (Very short answer) consists of 10 questions of 2 marks each. Maximum limit for each question is up to 40 words.

PART - II : (Short answer) consists of 5 questions of 4 marks each. Maximum limit for each question is up to 80 words.

PART - III : (Long answer) consists of 5 questions of 12 marks each with internal choice.

PART - I

1. Attempt **all** questions. Each question carries 2 marks. **10x2=20**

   (i) What is an algorithm ?

   (ii) Draw and list any 5 components used in a flow chart.

   (iii) Give the skeleton/basic outline of a C program.

   (iv) List logical and relational operators.

   (v) Give syntax of a while loop. Describe its features.

   (vi) Define an array. Declare an array to hold 5 real number values.

   (vii) What is a function prototype ? What are its elements ?

   (viii) What is a pointer ? Declare a pointer and an array and store the address of the array in the pointer.

   (ix) Describe using a diagram how the memory is allocated for each member of a structure.

   (x) How is a file opened for reading in 'read-only' mode ?

PART - II

2. Attempt **all** questions. Each question carries 4 marks. **5x4=20**

   (i) Write an algorithm to compute factorial of a number.

   (ii) Write a C program to check if the year entered is a Leap year or not. Leap year is defined as every 4^{th} year, if it is a non-century year, and every 400^{th} year, otherwise.

   (iii) Differentiate between for, while and do-while loops.

   (iv) What is recursion ? Write a recursive function to calculate HCF/GCD of two numbers.

   (v) Differentiate between structure and unions.

PART - III

3. What is : 4x3=12

(i) A Compiler

(ii) An Interpreter

(iii) An Assembler

(iv) A Linker

OR

Write pseudo-code and draw flow-chart to compute sum of digits of a positive integer. 12

4. Discuss about different operators available in C language. What is meant by operator precedence and associativity ? 8+4=12

OR

Write a C program using switch-case to print marks range given a student's grade as per the following table : 12

Grade Letter	Min. Marks	Max. Marks
D	0	40
C	40	60
B	60	80
A	80	100

5. Write a program to find all prime numbers between 1 and N. 12

OR

Write a C program to input and sort an array of integers using linear sort. 12

6. What is function definition ? Write a custom C function and use it in a program to find all occurrences of a character in a string. 5+7

OR

Write a program to transpose a matrix using custom function. Access the matrix using pointer notation. 12

7. (i) What are Structures and Unions ? How are they declared ? 4

(ii) Write a program to declare and use a structure to hold student data - roll no, name, program, and semester. Input details of 3 students and print them sequentially. 8

OR

Write a program to input multiple lines one-by-one and store them in a file. The input will end when the user types "STOP". Then read the file and print the output line by line again. 12

- o O o -