

**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY**  
**(Autonomous)**  
**Gandipet, Hyderabad -75**  
**Department Of Computer Science and Engineering**



**Course File**  
**PROGRAMMING FOR PROBLEM SOLVING**  
**Course Code: 20CSC01**  
**B.E. CSE-I Semester**

# **INDEX**

<b>S.No.</b>	<b>Topic</b>	<b>Pages</b>
1	Institute Vision and Mission	1
2	Department vision and mission	2
3	PEOs –PO & PSO's	3-5
4	CO-PO-PSO mapping theory	6
5	Course Outcomes – Syllabus	7-8
6	Lesson Plan	9-10
7	Theory Lecture schedule	11-220
8	Question Bank	221-259
9	Assignment	260-310
10	Class Test (1&2) - Key	311-323
11	End Exam Paper	324-325
12	End Exam Key	325-344
13	CIE	345-346
14	CO Attainment	347



**CHAITANYA BHARATHI  
INSTITUTE OF TECHNOLOGY (A)**  
Affiliated to Osmania University

## Institute Vision & Mission

### **Vision:**

To be a Centre of Excellence in Technical Education and Research

### **Mission:**

To address the emerging needs through quality technical education and advanced research



**CHAITANYA BHARATHI  
INSTITUTE OF TECHNOLOGY (A)**  
Affiliated to Osmania University

## Department of Computer Science

### **Vision:**

To be in the frontiers of Computer Science and Engineering with academic excellence and Research

### **Mission:**

The mission of Computer Science and Engineering Department is to:

1. Educate students with the best practices of Computer Science by integrating the latest research into the curriculum.
2. Develop professionals with sound knowledge in theory and practice of Computer Science and Engineering.
3. Facilitate the development of academia-industry collaboration and societal outreach programs.
4. Prepare students for full and ethical participation in a diverse society and encourage lifelong learning.



## Department of Computer Science

### **Program Education Objectives (PEOs):**

After the completion of the program, our:

1. Graduates will apply their knowledge and skills to succeed in their careers and/or obtain advanced degrees, provide solutions as entrepreneurs.
2. Graduates will creatively solve problems, communicate effectively, and successfully function in multi-disciplinary teams with superior work ethics and values.
3. Graduates will apply principles and practices of Computer Science, mathematics and Science to successfully complete hardware and/or software-related engineering projects to meet customer business objectives and/or productively engage in research.

### **Program Outcomes (PO)**

Engineering Graduates will be able to:

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem analysis:** Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/development of solutions:** Design solutions for complex engineering problems and design system components or processes that meet the specified needs

with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.

**4. Conduct investigations of complex problems:** Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

**5. Modern tool usage:** Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.

**6. The engineer and society:** Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.

**7. Environment and sustainability:** Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.

**8. Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

**9. Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

**10. Communication:** Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.

**11. Project management and finance:** Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.

**12. Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## **Program Specific Outcomes (PSOs):**

At the end of the program

1. Graduates will acquire the practical competency in Computer Science and Engineering through emerging technologies and open-source platforms related to the domains.
2. Graduates will design and develop innovative products by applying principles of computer science and engineering.
3. Graduates will be able to successfully pursue higher education in reputed institutions and provide solutions as entrepreneurs.
4. Graduates will be able to work in multidisciplinary teams for career growth by exhibiting work ethics and values.



## Department of Computer Science

**Course Code: 20CS C01**

**Course Title: PROGRAMMING FOR PROBLEM SOLVING**

**PROGRAMMING FOR PROBLEM SOLVING –THEORY (CO-PO-PSO MAPPING)**

	PO 1	P O 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	Po 10	PO 11	PO 12	PSO 1	PSO 2	PSO 3	PSO 4
20CS C01.1	2	1	2	-	-	-	-	-	-	1	1	2	2	2	-	-
20CS C01.2	3	3	3	2		1	-	-	-	1	1	2	2	3	2	2
20CS C01.3	2	2	2	1	1	-	-	-	-	1	-	2	2	2	2	1
20CS C01.4	3	3	3	2	3	-	-	-	-	1	1	2	2	3	2	1
20CS C01.5	3	2	2	1	2	-	-	-	-	1	-	2	3	3	2	1
20CS C01.6	3	3	-	-	3					3		3	3	3	3	2

**Instructor**

**20CS C01**

**20CS C01**

**PROGRAMMING FOR PROBLEM SOLVING**  
**(Common to All Programs)**

Instruction	3 Periods per week
Duration of End Examination	3 Hours
Semester End Examination	60 Marks
Sessional	40 Marks
Credits	3

**Course Objectives:** The objectives of this course are

1. Identification of computer components, Operating environments, IDEs.
2. Understanding the steps in problem solving and formulation of algorithms to problems.
3. Develop programming skills as a means of implementing a algorithmic solution with appropriate control and data structures.
4. Develop intuition to enable students to come up with creative approaches to problems.
5. Manipulation of text data using files.

**Course Outcomes:** On Successful completion of the course, students will be able to

1. Identify and understand the computing environments for scientific and mathematical problems.
2. Formulate solutions to problems with alternate approaches and represent them using algorithms / Flowcharts.
3. Choose data types and control structures to solve mathematical and scientific problem.
4. Decompose a problem into modules and use functions to implement the modules.
5. Apply arrays, pointers, structures, and unions to solve mathematical and scientific problems.
6. Develop applications using file I/O.

**UNIT -I**

**Introduction to computers and Problem Solving:** Components of a computer, Operating system, compilers, Program Development Environments, steps to solve problems, Algorithm, Flowchart / Pseudocode with examples.

**Introduction to programming:** Programming languages and generations, categorization of high-level languages.

**Introduction to C:** Introduction, structure of C program, keywords, identifiers, Variables, constants, I/O statements, operators, precedence, and associativity.

**UNIT – II**

**Introduction to decision control statements:** Selective, looping, and nested statements.

**Functions:** Introduction, uses of functions, Function definition, declaration, passing parameters to functions, recursion, scope of variables and storage classes, Case study using functions and control statements.

**UNIT – III**

**Arrays:** Introduction, declaration of arrays, accessing and storage of array elements, 1-dimensional array, Searching (linear and binary search algorithms) and sorting (Selection and Bubble) algorithms, 2-D arrays, matrix operations.

**Strings:** Introduction, strings representation, string operations with examples. Case study using arrays.

**UNIT – IV**

**Pointers:** Understanding computer's memory, introduction to pointers, declaration pointer variables, pointer arithmetic, pointers and strings, array of pointers, dynamic memory allocation, advantages, and drawbacks of pointers.

**Structures:** Structure definition, initialization and accessing the members of a structure, nested structures, structures and functions, self-referential structures, unions, and enumerated data types.

**UNIT-V**

**Files:** Introduction to files, file operations, reading data from files, writing data to files, error handling during file operations.

**Preprocessor Directives:** Types of preprocessor directives, examples.

**Textbooks:**

1. M.T. Somashekhar "Problem Solving with C", 2nd Edition, Prentice Hall India Learning Private Limited 2018
2. AKSharma "ComputerFundamentalsand Programming", 2nd Edition, University Press,2018
3. Pradeep Dey and Manas Ghosh, "ProgramminginC", OxfordPress,2nd Edition,2017

**Suggested Reading:**

1. Byron Gottfried, Schaum's Outline of Programming with C", McGraw- Hill.
2. Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Prentice Hall of India.
3. E. Balaguruswamy, Programming in ANSI C, Tata McGraw-Hill.
4. Reema Tharaja "Introduction to C Programming", Second Edition, OXFORD Press,2015.

**Online Resources:**

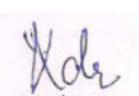
1. <https://www.tutorialspoint.com/cprogramming/index.htm>.
2. <https://onlinecourses.nptel.ac.in/noc18-cs10/preview>

<b>Lecture No</b>	<b>Topic</b>	<b>No. of classes</b>
<b>UNIT I</b>		
L.1	Introduction to Computers, Components of a computer, Operating system, compilers.	1
L.2	Program Development Environments, steps to solve problems.	1
L.3	Algorithm, Flowchart / Pseudo code with examples.	1
L.4	Introduction to Programming languages and generations, categorization of high level languages.	1
L.5	Introduction to C, structure of C program, keywords.	1
L.6	Identifiers, Variables, constants, I/O statements.	1
L.7	Operators, precedence and associativity.	1
L.8	Sample programs based on topics covered in this Unit.	1
<b>UNIT II</b>		
L.9	Introduction to decision control statements like If, If-Else.	1
L.10	Switch-Statement, Nested Statements and Examples.	1
L.11	Loop Control Statements: For, While, Do-While and Examples.	1
L.12	Continue, Break and Go To statements.	1
L.13	Case Study on control structures.	1
L.14	Basics of Functions, User-defined Functions.	1
L.15	Inter Function Communication, Standard Functions.	1
L.16	Parameter Passing-Call-by-value, call-by-reference, Recursion.	1
L.17	Storage Classes ( Auto, Register, Static, Extern), Scope Rules.	1
L.18	Case Study on Functions and Discussion of Previous Question Papers.	1
<b>UNIT III</b>		
L.19	Introduction to Arrays, Declaration, Using Arrays in C.	1
L.20	Accessing and storage of array elements, one dimensional arrays.	1

L.21	Searching Algorithms (linear and binary search).	1
L.22	Sorting Algorithms (Selection and Bubble).	1
L.23	Two dimensional arrays, matrix operations.	1
L.24	Introduction to Strings, Strings representation.	1
L.25	String operations with examples.	1
L.26	Case Study on arrays and Discussion of Previous Question Papers.	1
<b>UNIT IV</b>		
L.27	Introduction to Pointers, Pointer declaration, Pointer Variables.	1
L.28	Arrays and Pointers, Pointer Arithmetic and Arrays.	1
L.29	Pointers and strings.	1
L.30	Array of pointers, Dynamic memory allocation.	1
L.31	Advantages and drawbacks of pointers and Discussion of Previous Question Papers.	1
L.32	<b>Structures:</b> Definition and Initialization of Structures.	1
L.33	Accessing Members of Structures, Nested Structures.	1
L.34	Self Referential Structures.	
L.35	Structures and Functions.	1
L.36	Unions and Enumerated Types.	1
<b>UNIT V</b>		
L.37	Introduction to files, file operations.	1
L.38	Reading data from files.	1
L.39	Writing data to files, error handling during file operations.	1
L.40	Preprocessor Directives: Types of preprocessor directives, examples.	1
L.41	Revision of previous question papers and Key Points/ Topics.	1



Instructor



Head, Department of CSE



# CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. [www.cbit.ac.in](http://www.cbit.ac.in)



COMMITTED TO  
RESEARCH,  
INNOVATION AND  
EDUCATION

**41**  
years

**Course Code:** 20CS C01

**Course Title:** PROGRAMMING FOR PROBLEM SOLVING

## UNIT-1

### Objective:

The primary goals of the module/unit are to introduce:

- Basic representation of data and how to process data using binary representation
- Inside a computer.
- Techniques for specifying data, operations on data, and problem solving using a Programming language.
- Identification of computer components, Operating environments, IDEs.
- Categorization of high-level languages.
- To write an Algorithm and flowchart for knowing steps to solve a problem.
- Usage of keywords, identifiers, operators to solve mathematical questions.

### Outcome:

On Successful completion of the course, students will be able to Identify and understand the computing environments for scientific and mathematical problems.

### Scope:

The Module/Unit covers the following topics: Components of a computer, Operating system, compilers, Program Development Environments, steps to solve problems, Algorithm, Flowchart / Pseudocode with examples. Programming languages and generations, categorization of high-level

languages. Structure of C program, keywords, identifiers, Variables, constants, I/O statements, operators, precedence, and associativity.

While the topics are taught by introducing programming development environments and also how to write algorithm and flowchart for to solve problems using C language

**Text and Reference:**

**(a) Text Book:**

**TB1.** Manas Ghosh, Pradip Dey ,” Programming in C “ 2nd Edition Oxford university press.

**TB2.** M.T. Somashekhar “Problem Solving with C”, 2nd Edition, Prentice Hall India Learning Private Limited 2018

**(b) Reference Books:**

**RB1.** Byron Gottfried ,Schaum’s ”Outline of Programming with C”, McGraw- Hill

<b>Module /Lecture</b>	<b>Theme</b>	<b>Learning Objectives</b>
I	Basic data, data types, and data representation	To understand how to define, represent, and process basic data.
II	Analyzing, designing, and managing a process or program for any given problem.	To diagrammatically understand and visualize an algorithm using boxes of various kinds. This representation gives a step by step insight to a solution of a given problem.
III	Writing algorithm and flowchart to solve any given problem.	By writing algorithm and representing in flowchart helps student to solve a given problem.
IV	Operators and usage of keywords	Can know the working of different operators and its associativity and precedence.

V	I/O statements in header file.	Working of basic input and output statements using stdio header file.
---	--------------------------------	---

( Note: Take Lecture No, Theme from lesson plan )

Lecture title	( L1 etc)
Date	[ You Can Write During the Course]

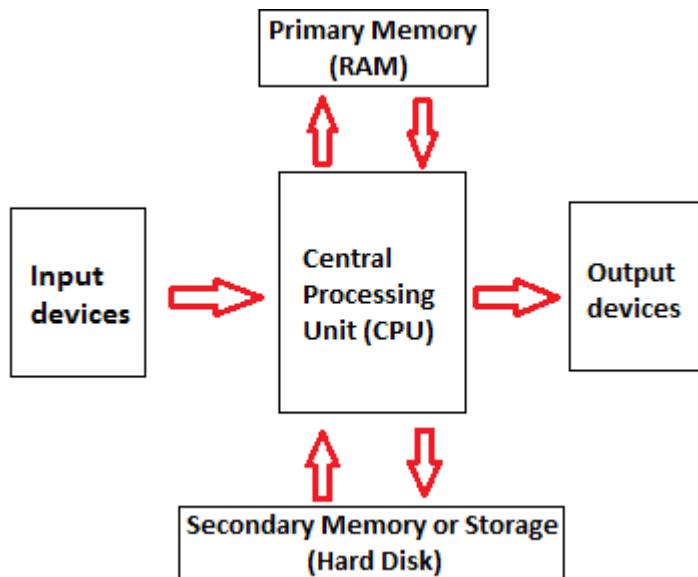
### Key topics / themes of the lecture :

#### Notes, comments and your own Examples with answers/ questions

#### Components of a computer:

There are 5 main computer components that are given below:

- Input Devices
- CPU
- Output Devices
- Primary Memory
- Secondary Memory



The operations of computer components are given below:

- 1) **Inputting:** It is the process of entering raw data, instructions and information into the computer. It is performed with the help of input devices.

**2) Storing:** The computer has primary memory and secondary storage to store data and instructions. It stores the data before sending it to CPU for processing and also stores the processed data before displaying it as output.

**3) Processing:** It is the process of converting the raw data into useful information. This process is performed by the CPU of the computer. It takes the raw data from storage, processes it and then sends back the processed data to storage.

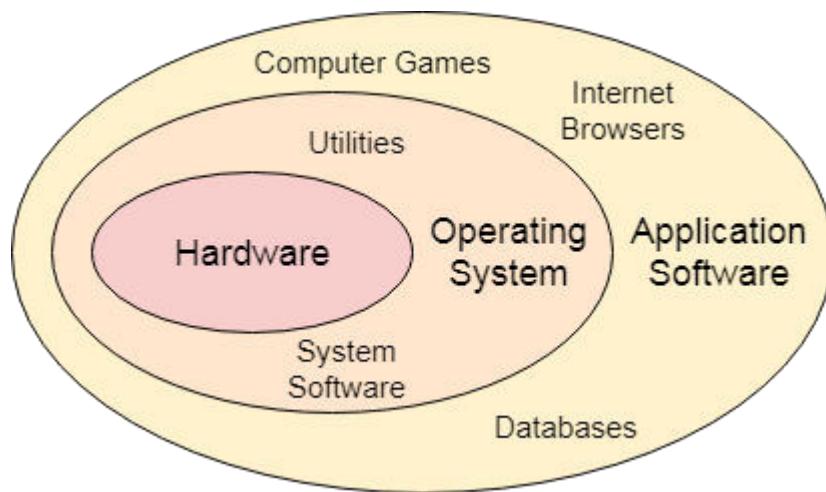
**4) Outputting:** It is the process of presenting the processed data through output devices like monitor, printer and speakers.

**5) Controlling:** This operation is performed by the control unit that is part of CPU. The control unit ensures that all basic operations are executed in a right manner and sequence.

### **Operatingsystem:**

In the Computer System (comprises of Hardware and software), Hardware can only understand machine code (in the form of 0 and 1) which doesn't make any sense to a naive user.

We need a system which can act as an intermediary and manage all the processes and resources present in the system.



An **Operating System** can be defined as an **interface between user and hardware**. It is responsible for the execution of all the processes, Resource Allocation, CPU management, File Management and many other tasks.

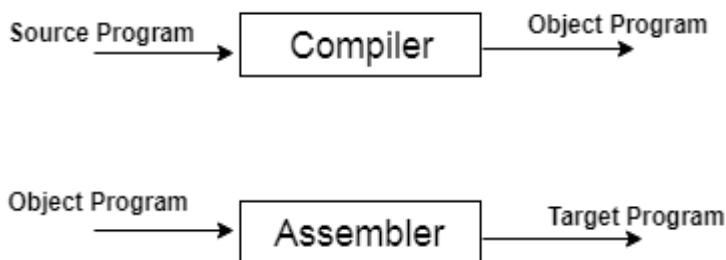
The purpose of an operating system is to provide an environment in which a user can execute programs in convenient and efficient manner.

### **Compilers:**

- A compiler is a translator that converts the high-level language into the machine

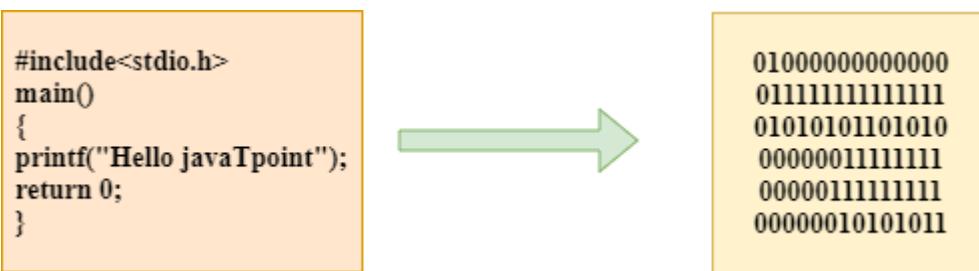
language.

- High-level language is written by a developer and machine language can be understood by the processor.
- Compiler is used to show errors to the programmer.
- The main purpose of compiler is to change the code written in one language without changing the meaning of the program.
- When you execute a program which is written in HLL programming language then it executes into two parts.
- In the first part, the source program compiled and translated into the object program (low level language).
- In the second part, object program translated into the target program through the assembler.



### Compilation Process in C:

The compilation is a process of converting the source code into object code. It is done with the help of the compiler. The compiler checks the source code for the syntactical or structural errors, and if the source code is error-free, then it generates the object code.



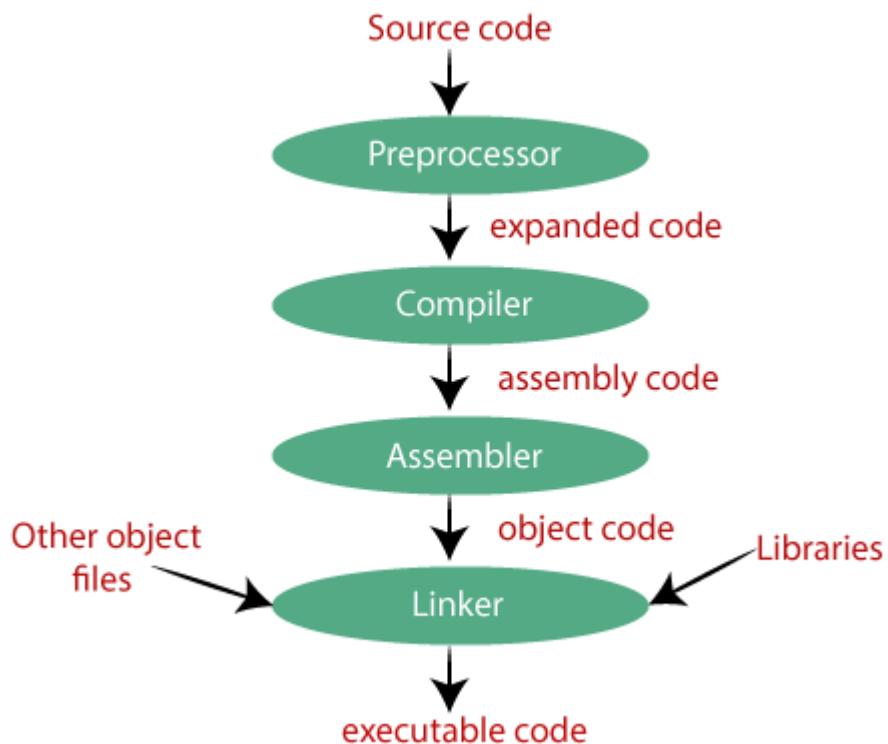
The c compilation process converts the source code taken as input into the object code or machine code. The compilation process can be divided into four steps, i.e., Pre-processing, Compiling, Assembling, and Linking.

The preprocessor takes the source code as an input, and it removes all the comments from the source code. The preprocessor takes the preprocessor directive and interprets it. For example, if `<stdio.h>`, the directive is available in the program, then the preprocessor interprets the

directive and replace this directive with the content of the '**stdio.h**' file.

The following are the phases through which our program passes before being transformed into an executable form:

- **Preprocessor**
- **Compiler**
- **Assembler**
- **Linker**



### Preprocessor

The source code is the code which is written in a text editor and the source code file is given an extension ".c". This source code is first passed to the preprocessor, and then the preprocessor expands this code. After expanding the code, the expanded code is passed to the compiler.

### Compiler

The code which is expanded by the preprocessor is passed to the compiler. The compiler converts this code into assembly code. Or we can say that the C compiler converts the pre-processed code into assembly code.

## Assembler

The assembly code is converted into object code by using an assembler. The name of the object file generated by the assembler is the same as the source file. The extension of the object file in DOS is '.obj,' and in UNIX, the extension is 'o'. If the name of the source file is '**hello.c**', then the name of the object file would be 'hello.obj'.

## Linker

Mainly, all the programs written in C use library functions. These library functions are pre-compiled, and the object code of these library files is stored with '.lib' (or '.a') extension. The main working of the linker is to combine the object code of library files with the object code of our program. Sometimes the situation arises when our program refers to the functions defined in other files; then linker plays a very important role in this. It links the object code of these files to our program. Therefore, we conclude that the job of the linker is to link the object code of our program with the object code of the library files and other files. The output of the linker is the executable file. The name of the executable file is the same as the source file but differs only in their extensions. In DOS, the extension of the executable file is '.exe', and in UNIX, the executable file can be named as 'a.out'. For example, if we are using printf() function in a program, then the linker adds its associated code in an output file.

## Program Development Environments:

The development environment helps the developers to develop the application or product using a set of processes and programming tools.

An development environment provides developers an interface and convenient view of the development process which includes writing code, testing the same and packaging the build so that it can be deployed.

Ex:

Microsoft Visual Studio

Turbo C/C++

Dev C/C++

## Flowchart:

**Flowchart** is a diagrammatic representation of sequence of logical steps of a program. Flowcharts use simple geometric shapes to depict processes and arrows to show relationships and process/data flow.

## Flowchart Symbols

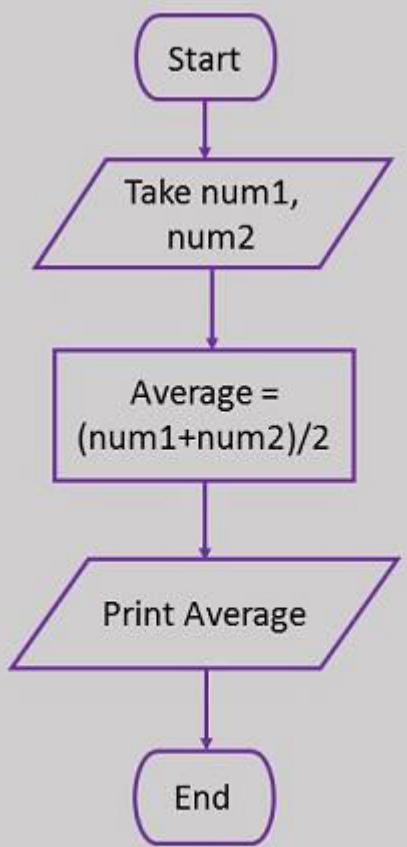
Here is a chart for some of the common symbols used in drawing flowcharts.

Symbol	Symbol Name	Purpose
--------	-------------	---------

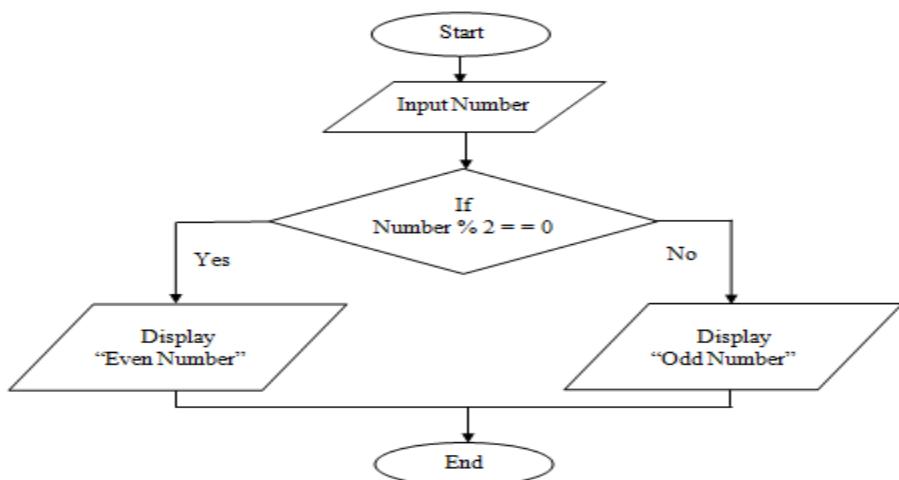
	Start/Stop	Used at the beginning and end of the algorithm to show start and end of the program.
	Process	Indicates processes like mathematical operations.
	Input/ Output	Used for denoting program inputs and outputs.
	Decision	Stands for decision statements in a program, where answer is usually Yes or No.
	Arrow	Shows relationships between different shapes.
	On-page Connector	Connects two or more parts of a flowchart, which are on the same page.
	Off-page Connector	Connects two parts of a flowchart which are spread over different pages.

Examples:

Here is a flowchart to calculate the average of two numbers.



Here is a flowchart to calculate the even or odd number.



## Programming Language

As we know, to communicate with a person, we need a specific language, similarly to communicate with computers, programmers also need a language is called Programming language.

Before learning the programming language, let's understand what is language?

### What is Language?

Language is a mode of communication that is used to **share ideas, opinions with each other**. For example, if we want to teach someone, we need a language that is understandable by both communicators.

### What is a Programming Language?

A programming language is a **computer language** that is used by **programmers (developers) to communicate with computers**. It is a set of instructions written in any specific language ( C, C++, Java, Python) to perform a specific task.

A programming language is mainly used to **develop desktop applications, websites, and mobile applications**.

### Types of programming language

#### 1. Low-level programming language

Low-level language is **machine-dependent (0s and 1s)** programming language. The processor runs low- level programs directly without the need of a compiler or interpreter, so the programs written in low-level language can be run very fast.

Low-level language is further divided into two parts -

##### i. Machine Language

Machine language is a type of low-level programming language. It is also called as **machine code or object code**. Machine language is easier to read because it is normally displayed in binary or hexadecimal form (base 16) form. It does not require a translator to convert the programs because computers directly understand the machine language programs.

The advantage of machine language is that it helps the programmer to execute the programs faster than the high-level programming language.

##### ii. Assembly Language

Assembly language (ASM) is also a type of low-level programming language that is designed for

specific processors. It represents the set of instructions in a **symbolic and human-understandable form**. It uses an assembler to convert the assembly language to machine language.

The advantage of assembly language is that it requires less memory and less execution time to execute a program.

## 2. High-level programming language

High-level programming language (HLL) is designed for **developing user-friendly software programs and websites**. This programming language requires a compiler or interpreter to translate the program into machine language (execute the program).

The main advantage of a high-level language is that it is **easy to read, write, and maintain**.

High-level programming language includes **Python, Java, JavaScript, PHP, C#, C++, Objective C, Cobol, Perl, Pascal, LISP, FORTRAN, and Swift programming language**.

A high-level language is further divided into three parts -

### i. Procedural Oriented programming language

Procedural Oriented Programming (POP) language is derived from structured programming and based upon the procedure call concept. It divides a program into small procedures called **routines or functions**.

Procedural Oriented programming language is used by a software programmer to create a program that can be accomplished by using a programming editor like IDE, Adobe Dreamweaver, or Microsoft Visual Studio.

The advantage of POP language is that it helps programmers to easily track the program flow and code can be reused in different parts of the program.

*The advantage of POP language is that it helps programmers to easily track the program flow and code can be reused in different parts of the program.*

**Example:** C, FORTRAN, Basic, Pascal, etc.

### ii. Object-Oriented Programming language

Object-Oriented Programming (OOP) language is **based upon the objects**. In this **programming language, programs are divided into small parts called objects**. It is used to implement real-world entities like inheritance, polymorphism, abstraction, etc in the program to makes the program resusable, efficient, and easy-to-use.

The main advantage of object-oriented programming is that OOP is faster and easier to execute,

maintain, modify, as well as debug.

*Note: Object-Oriented Programming language follows a bottom-up approach.*

**Example:** C++, Java, Python, C#, etc.

### iii. Natural language

Natural language is a **part of human languages** such as English, Russian, German, and Japanese. It is used by machines to understand, manipulate, and interpret human's language. It is used by developers to **perform tasks such as translation, automatic summarization, Named Entity Recognition (NER), relationship extraction, and topic segmentation.**

The main advantage of natural language is that it helps users to ask questions in any subject and directly respond within seconds.

## 3. Middle-level programming language

Middle-level programming language **lies between the low-level programming language and high-level programming language.** It is also known as the intermediate programming language and pseudo-language.

A middle-level programming language's advantages are that it supports the features of high-level programming, it is a user-friendly language, and closely related to machine language and human language.

**Example:** C, C++ language

### Introduction to C:

The C Language is developed by Dennis Ritchie for creating system applications that directly interact with the hardware devices such as drivers, kernels, etc.

C programming is considered as the base for other programming languages, that is why it is known as mother language.

It can be defined by the following ways:

1. Mother language
2. System programming language
3. Procedure-oriented programming language
4. Structured programming language
5. Mid-level programming language

**structure of C program:**

Documentation section

Link section

Definition section

Global declaration section

main () Function section

{

Declaration part

Executable part

}

Subprogram section

Function 1

Function 2

.....

.....

Function n

(User defined functions)

**Documentation Section :**

This section consists of a set of comment lines useful for documentation. Though comments are not necessary in a program, they help users to understand the program better.

**Link(Preprocessor) and Definition Section:**

This section contains header files which begin with a # symbol and are extended with .h. They are also known as preprocessor directives used to include header files in a C program. Examples of preprocessor statements are

#include<stdio.h>

#include<math.h>

Symbolic Constants are included using # define. For example to define the value of PI, the statement is:

#define PI 3.14159

**Global Declaration Section:**

Global declaration section is used to define variables that would be used in more than one function. Such variables are called global variables and they must be declared before the main( ) function.

### **The main() Function:**

All C programs must contain main( ) function. Further there must be only one main( ) function. main( ) denotes the starting of a C program.

### **Braces :**

All C programs incorporate a set of curly braces { }. Execution of the program begins at the opening brace { of the main( ) function and ends at its closing brace }.

### **Declaration Part:**

This part is used to declare all the variables, arrays functions etc. used in the C program. Initialization along with declaration can also be done here.

### **Executable Part :**

This part of the program consists of a set of executable statements such as Input/ Output statements, arithmetic statements, control statements etc. It can also include comment statements which are ignored by the compiler (non-executable). The declaration part and the executable part must be within opening and closing braces.

Every statement in the declaration part and executable part ends with a semicolon.

### **Subroutine Section:**

This section is optional and consists of all the user-defined functions called in the main( ) function.

### **Example:**

```
/*Program to find the area of a square*/
#include <stdio.h>
main()
{
float side, area ;
side=5;
area=side*side;
printf("\nArea of a square=%f",area);
}
```

Output:  
Area of a square = 25.0

### **C KEYWORDS:**

There are some reserved words in C, called keywords. All the keywords have standard pre-defined meanings and can be used only for the purpose intended. All keywords must be written in lowercase. They cannot be used as user-defined identifiers. The standard keywords are listed below

auto	break	case	char	const	continue	default	do	double	else
------	-------	------	------	-------	----------	---------	----	--------	------

enum	extern	float	for	goto	if	int	longregister	return
short	signed	sizeof	static	struct	switch	typedef	union	unsigned
volatile	while							void

### **IDENTIFIERS:**

- Identifiers refer to the names of program elements such as variables, functions and arrays.
- Identifiers are sequence of characters chosen from the set A–Z, a–z, 0–9, and \_ (underscore).
- C is a case sensitive language so that ALFA and Alfa are different.
- The underscore symbol \_ is generally in the middle of an identifier (embedded).
- Identifiers may be of reasonable length of 8–10 characters though certain computers allow up to 32 characters.
- Identifier names must start with an alphabet or underscore followed by letters, digit or a combination of both.
- C has a list of keywords that cannot be used anywhere other than that predefined in the language i.e., you can't have a variable named int. Also one must follow the practice of choosing names for variables which indicate the roles of those variables in the program.

### **VARIABLES:**

A variable is an identifier used to store a single data item. This data could be a numerical quantity or a character constant. During execution of the program, data must be assigned to the variable. Therefore a variable can take different values at different times during the execution of the program. The data stored in the variable can be accessed any time, by referring to the variable name.

The rules for naming a variable are the same as those for an identifier.

They are :

1. Variable names generally begin with a letter followed by letters, digits underscore or a combination of all.
2. No special character except the underscore symbol ( \_ ) is allowed. Underscore is generally embedded in a variable name to make it more readable.
3. Maximum length of a variable name should not exceed 32 characters. Normally the length must not exceed 8 characters since many compilers treat the first 8 characters as significant.
4. C variables are case sensitive, so that Counter, counter and COUNTER are 3 different variables.
5. C keywords cannot be used as variable names.
6. White spaces are not allowed in a variable name.
7. Appropriate variable names must be chosen to indicate their role in the program.

### **CONSTANTS:**

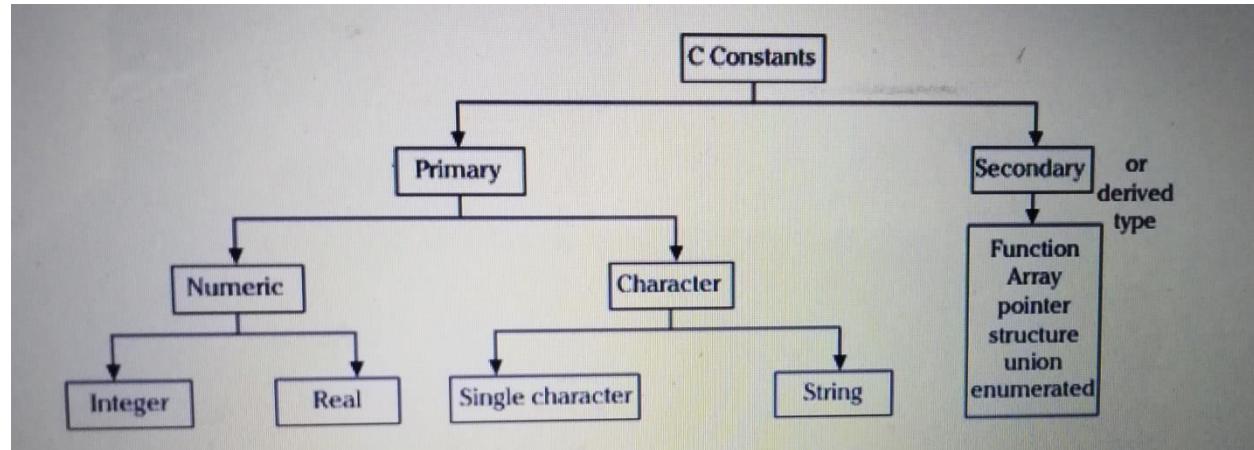
Any fixed value that does not change during the execution of a program is known as a constant. C consists of several types of constants.

### **Types of C constants:**

C constants can be divided into two categories :

- Primary Constants
- Secondary Constants

These constants can further be divided as follows :



### **Integer Constants :**

Integers and real constants are numbers and are generally referred to as numeric constants.

An integer constant consists of a sequence of digits and is an integer-valued number. There are 3 types of integer constants depending on the number system.

They are:

- Decimal (base 10)
- Octal (base 8)
- Hexadecimal (base 16)

### **Real Constants:**

Quantities which are represented by numbers with fractional part are called real or floating point constants.

- (a) A real constant must have at least 1 digit.
- (b) It must have a decimal point.
- (c) It can be either positive or negative.
- (d) No commas or blanks are allowed within a real constant.

### **Character Constants:**

- (a) A character constant is a single character within single quotes.
- (b) The maximum length of a character constant is 1.
- (c) Arithmetic operations are possible on character constant since they too represent integer values.
- (d) C also recognizes all the backlash character constants (Escape sequences) available.

Note that the character constant ‘2’ is not the same as the number 2. Every character has an equivalent integer value known as the ASCII code. Some character constant & their equivalent ASCII values are as below :

‘A’ = 65

‘B’ = 66

‘a’ = 97

‘0’ = 48

‘1’ = 49

### **String Constants:**

A string constant consists of zero or more number of characters enclosed within double quotes.

The characters within quotes could be digits, characters, special characters and blank spaces.

Ex: "Red", "Rs.20.25", " "

Note that ‘A’ is not equal to “A” since a string constant containing a single character does not have a corresponding integer value. In fact it contains two characters, the specified character followed by a null character represents by \0.

### **Coding constants:**

There are 3 types of code constants used in programs:

- Literal constants
- Defined constants
- Memory constants

**Literal constants:** are used to specify data, which cannot be changed.

For example: x = y + 1 ;

In this statement 1 is a literal constant.

**Defined Constants:** are constants that can be defined using a preprocessor command #define.

This constant is placed at the beginning of a program since it is a preprocessor command.

For example :

# define MAX 30

The defined constants make the program easily modifiable and enhance the understandability of the program.

**Memory Constants:** use a c type qualifier called const, to indicate that the data specified is a constant and that it cannot be changed.

Its general format is :

const Datatype identifier = value ;

For example :

const float pi = 3.14159 ;

defines a memory constant pi of float type with a value of 3.14159.

### **I/O statements:**

In C programming you can use `scanf()` and `printf()` predefined function to read and print data.

### i) **printf**

This function is used for displaying the output on the screen i.e the data is moved from the computer memory to the output device.

#### **Syntax:**

```
printf("format string", arg1, arg2, ....);
```

In the above syntax, 'format string' will contain the information that is formatted. They are the general characters which will be displayed as they are .  
arg1, arg2 are the output data items.

#### **Example:** Demonstrating the printf function

```
printf("Enter a value:");
```

**printf** will generally examine from left to right of the string.

The characters are displayed on the screen in the manner they are encountered until it comes across **% or \**.

Once it comes across the conversion specifiers it will take the first argument and print it in the format given.

### ii) **scanf**

**scanf** is used when we enter data by using an input device.

#### **Syntax:**

```
scanf ("format string", &arg1, &arg2, ....);
```

The number of items which are successful are returned.

Format string consists of the conversion specifier. Arguments can be variables or array name and represent the address of the variable. Each variable must be preceded by an ampersand (&). Array names should never begin with an ampersand.

#### **Example:** Demonstrating scanf

```
int avg;  
float per;  
char grade;  
scanf("%d %f %c",&avg, &per, &grade);
```

**scanf** works totally opposite to **printf**. The input is read, interpret using the conversion specifier and stores it in the given variable.

The conversion specifier for **scanf** is the same as **printf**.

**scanf** reads the characters from the input as long as the characters match or it will terminate.

The order of the characters that are entered are not important.

It requires an enter key in order to accept an input.

### **OPERATORS IN C :**

C provides several operators for elementary arithmetic operations like addition, subtraction, multiplication, division, residue-modulo (the operation that gives the remainder after division of one integer by another) and logical manipulations.

An operator is a symbol used for certain type of manipulations, logical or mathematical. The data on which the operators act are called operands. An expression is a valid combination of operators and operands that results in a value.

For example in the expression : 12\*5

\* is the multiplication operator,

12 and 5 are the operands.

Operators may be classified into 4 categories namely

- (i) unary
- (ii) Binary
- (iii) Ternary
- (iv) Special

#### **Unary Operators:**

Operators that operate on a single operand are known as unary operators. The unary operators always precede the operand. The different unary operators available in C are :

- (i) Unary plus (+)
- (ii) Unary minus (-)
- (iii) Increment (++)
- (iv) Decrement (--)
- (v) Logical NOT (!)
- (vi) Bitwise complement (~)
- (vii) Address (&)

The unary minus operator is used for negation. It changes sign of the quantity on its right i.e., it multiplies its only operand by -1. Similarly unary plus multiplies its operand by +1.

#### **Binary Operators:**

Binary operators operate on two operands. They are classified into five categories as follows:

- (i) Arithmetic
- (ii) Relational
- (iii) Logical
- (iv) Assignment
- (v) Bitwise

### **Ternary Operators:**

Ternary operators act on three data elements. The conditional operator (?:) is a ternary operator

### **Special Operators:**

Special operators include operators such as comma (,) and sizeof.

### **ARITHMETIC OPERATORS:**

C supports five arithmetic operators.

<b>Operator</b>	<b>Purpose</b>
+	Addition
-	Subtraction or (unary)
*	Multiplication
/	Division
%	Remainder after integer division

There is no exponentiation operator available in C. However, the library function pow( ) is provided to compute exponentiation.

Dividing one integer by another integer is called as integer division, which truncates, the fractional part. The % operator gives the remainder of such a division.

For example, consider  $a = 10$ ,  $b = 3$ .

$a \% b$  gives  $10 \% 3$  giving the result 1

and  $a/b$  gives  $10/3$  giving the result 3

% operator can have only integer operands.

Mixed-mode Arithmetic operations consists of a real and an integer operand. If one of the operand is real, the result is also real.

For example :

$15/2.0 = 7.5$

However  $15/2 = 7$

### **RELATIONAL OPERATORS:**

The relational operators are used to form relational expressions, representing conditions that are either true or false. The result of the expression is an integer. Further, false is represented by zero value and true is represented by non-zero value. For example expressions

$50 < 200$

$x > y$

are called relational expressions whose value can be either true or false i.e.,  $50 < 200$  is true. when  $x = 10$  and  $y = 5$ ,  $x > y$  is true. There are 6 relational operators in C.

They are:

<b>Operator</b>	<b>Meaning</b>
>	greater than
<	Less than
$\geq$	Greater than or equal to
$\leq$	Less than or equal to
$=$	Equal to
$\neq$	Not equal to

Consider 3 integer variables,  $i = 1$ ,  $j = 2$  and  $k = 7$

<b>Expression</b>	<b>Logical value</b>	<b>Integral value</b>
(i) $10.5 \leq 12$	true	1
(ii) $i == 20$	false	0
(iii) $5 > 20$	false	0
(iv) $(i+j) \geq k$	false	0
(v) $(j+k+1) > (i+5)$	true	1
(vi) $k != 7$	false	0

In the 4th and 5th example given above the expressions contain arithmetic expression. Thus in the relational expression  $(i+j) \geq k$ , the arithmetic expression  $i+j$  will be evaluated and then the result is compared with  $k$ . These arithmetic operators have higher priority over relational operators. The relational operators are used to compare two quantities in decision statements.

### **LOGICAL OPERATORS:**

Logical operators are used to combine two or more logical expressions. There are 3 logical operators in C. They are logical AND, logical OR and logical NOT

<b>Operator</b>	<b>Meaning</b>
!	logical NOT
$\&\&$	logical AND
$\ $	logical OR

- The result of a logical AND is true only if both operands are true. The result is false if either operand is false.
- The result of a logical OR is false only if both operands are false. The result is true if either operand is true.
- C also consists of the unary operator ! (not) that negates the value of a logical expression i.e., it causes true expression to become false and vice-versa. This operator is known as logical NOT operator. Further like relational operators, a logical expression results in zero or non-zero value depending on whether it is false or true.

Consider  $i = 8$ ,  $x = 5.1$ ,  $c = 'a'$  where  $i$  is an integer variable,  $x$  is a floating point variable and  $c$  is a char variable.

<b>Expression</b>	<b>Logical value</b>	<b>Integer value</b>
(i>6)&&(c=='a')	true	1 (or non-zero)
i<=6  (c=='b')	false	0
i>=6&&(c==97)	true	1
!(i<=4)	true	1

## ASSIGNMENT OPERATORS

The assignment operator '=' is used to assign the value of a variable, constant or expression to a variable. The syntax is :

**variable=variable/constant/expression;**

For example :

```
x=10;      /*x is assigned a value 10 */

b=x;      /*The value of x is assigned to b i.e.,10*/

sum=a+b;  /*The sum of a&b which is 20 is assigned to sum*/
```

C also provides abbreviated or shorthand assignment operators. They are of the form:

**Variable operator = Expression;**

For example the statement

**n = n + 5;**

can be written using shorthand assignment operator as :

**n += 5;**

Where += is the abbreviated or compound operator. In the above case += means add 5 to n. The different shorthand assignment operators are:

<b>Normal assignment</b>	<b>Shorthand assignment</b>	<b>Operator</b>	<b>Meaning</b>
n = n + 1	n += 1	+=	sum & assign
n = n - 50	n -= 50	-=	subtract & assign
n = n * 10	n *= 10	*=	multiply & assign
n = n / 6	n /= 6	/=	divide & assign
n = n % 5	n %= 5	%=	find modulo and assign

The priority of the above operators `+=`, `-=`, `*=`, `/=` and `%=` is the same as the assignment operator.

### Difference between `==` and `=` operator

1. `==` is an equality or relational operator, used for comparison `=` is an assignment operator which assigns the resultant value on its right hand side to the variable on its left hand side.
2. `==` operator does not change the value of the variable on the left hand side. However `=` operator does change the value of the variable on the left hand side of the assignment statement.

## INCREMENT/DECREMENT OPERATOR

The increment and decrement operators are specifically used in C or C++.

They are not generally available in other languages.

Increment operator `++` is used to increment the value of the variable by 1. Decrement operator `--` is used to decrement the value of the variable by 1.

They are also called as unary operators Fig. 2.14. Since they operate on only one data element.

Operator	Meanings	Usage
<code>++</code>	Increment	<code>a++</code> or <code>++ a</code> means $a = a + 1$
<code>--</code>	Decrement	<code>a--</code> or <code>-- a</code> means $a = a - 1$

Consider the following statements :

```
int x,y;  
:  
x = 10;  
y = ++x;
```

`x` is incremented by 1 so that `x` and `y` will have a value 11. Similarly in

```
int a, b;  
:  
a = 5;  
b = --a;
```

will result in the value of `a` and `b` to be 4.

**Post-increment operator `++`** is written to the right of its operand as in `n ++`; which is the

same as  $n = n + 1;$

**Post-decrement operator**  $--$  is also written to the right of its operand as in  $n--$ , and is equivalent to  $n = n - 1;$

Pre-increment and pre-decrement **operators**  $++$  and  $--$  are always placed to the left of the operand as follows :

```
++n; /* same as n = n+1*/  
-n; /* same as n = n-1*/
```

The behavior of post and pre-increment or decrement operators are the same when used with a single operand. However, they behave differently when the operand is embedded in an expression.

### Difference between postfix & prefix operators

- (i) The pre-increment or decrement operator first increments or decrements the value of its operand by 1 and then assigns the resulting value to the variable on the left hand side. For example, consider

```
int a, b; a =  
15;  
b = ++a;
```

In this example, the value of  $a$  is incremented by 1 and then this value 16 is assigned to  $b$ .

- (ii) The post increment or decrement operator first assigns the value to the variable on the left hand side and then increments or decrements the operand by 1. Consider the following program segment:

```
int m, n; n =  
2;  
m = n--;
```

The value of  $n$  is first assigned to  $m$  and then it is decremented i.e.,  $n$  becomes 1, but value of  $m$  is 2.

In subscripted variables.

```
a[j++]=5;
```

is equivalent to  $a[j]=5;$

```
j++;
```

Similarly,  $a[++i]=5;$

is equivalent to  $i = i+1;$

```
a[i] = 5;
```

Consider the statement

```
x = n++/2;
```

where n = 5. After execution, n is 6 but x gets the value 5/2 which is 2.

## TERNARY OPERATOR (CONDITIONAL OPERATOR)

C consists of a very special operator known as ternary or conditional operator and is represented by ?: The syntax is

```
expression1 ? expression2 : expression3;
```

and general usage is as

```
variable = expression1 ? expression2 : expression3;
```

Since it contains 3 expressions it is called ternary. The above statement is evaluated as follows :

- (i) Expression1 is evaluated first.
- (ii) If it is true or non-zero, expression2 is evaluated and its value assigned to the variable.
- (iii) If expression1 is false or zero, expression3 is evaluated and its value assigned to the variable.
- (iv) Either expression2 or expression3 will be evaluated, but never both.

Consider the following statements :

```
n = 5;  
m = 10;  
big = (n > m) ? n : m;
```

Here n > m is evaluated first giving the result ‘false’. Since the expression is false the value of m is assigned to big. Thus big = 10. The same can be written in the form of if else statement as follows :

```
if(n > m) big = n; else  
big = m;
```

Some more examples :

<pre>(10 == 8) ? 2:4</pre>	<pre>//result is 4 since 10 is not equal to 8</pre>
<pre>(10&gt;8) ? 2:4</pre>	<pre>//result is 2 since 10 is greater than 8</pre>
<pre>(10==8+3) ? 2:4</pre>	<pre>//result is 4 since 10 is not equal to 11</pre>

## SPECIAL OPERATORS

C consists of some special operators. The operators `,`, `&`, `*`, `.` and `sizeof`, are called special operators in C.

Operator	Description
<code>,</code>	<b>Comma operator</b>
<code>sizeof</code>	<b>size in bytes</b>
<code>&amp;</code>	<b>address of operand</b>
<code>*</code>	<b>accessing value at that address</b>
<code>.</code>	<b>dot operator</b>
<code> </code>	<b>arrow operator</b>

### The Comma Operator

- (i) The comma operator is used to separate a list of expressions given as a single expression.
- (ii) The lists with commas are always evaluated from left to right.
- (iii) The value of the right most expression becomes the final value of the combined expression.
- (iv) The syntax is of the form:

```
variable =exp1,exp2,..... expn;
```

For example, the statement:

```
result = (a = 5, b = 10, a + b);
```

- ◆ first assigns 5 to a i.e., a=5
- ◆ next assigns 10 to b i.e., b=10
- ◆ and finally assigns the value of a+b i.e., 10+5=15 to result.

- (v) The comma operator needs parentheses, since it has the lowest precedence compared to all other operators.
- (vi) The comma operator is used in for( ) loops to separate more than one expression. It

```
for (exp1a, exp1b; exp2; exp3a, exp3b)statement;
```

is possible to write:

- (vii) It can also be used in while loops to separate more than one expression. For example:

```
while(textchar=getchar( ),textchar != '\n')
putchar(textchar);
```

In this example, the end condition for input is checked first and then it waits for input of a character. The program terminates as soon as the enter key is pressed.

### The sizeof operator

- (i) The sizeof operator returns the size in bytes of the operand on the right.
- (ii) The syntax is

**sizeof(n)**

where n can be a variable, a constant or a data type.

- (iii) It is a unary operator which associates from right to left.
- (iv) The size of operator is used to determine the size of an array or a structure. It is also used for dynamic memory allocation of variables.

### Example 2.30

```

/* Program to show usage of sizeof operator
 */
#define STRING "Usage of sizeof
Operator" #include <stdio.h>
main()
{
char a = 'p';
clrscr();
printf("\n The size of char is : %d \n", sizeof(char));
printf("\n Therefore the size of char is : %d \n", sizeof(a));
printf("\n However the size of 'p' is : %d \n", sizeof('p'));
printf("\n STRING : \"%s\" \n", STRING);
printf("\n The number of bytes in STRING is : %d
\n", sizeof STRING); printf("\n The size of short is : %d\n", sizeof(short));
printf("\n The size of int is : %d\n", sizeof(int));
printf("\n The size of long is : %d\n", sizeof(long));
printf("\n The size of float is : %d\n", sizeof(float));
printf("\n The size of double is : %d
\n", sizeof(double)); getch();
}

```

<b>Output</b>	
<b>The size of char is</b>	<b>1</b>
<b>Therefore the size of char is</b>	<b>1</b>
<b>However the size of 'p' is</b>	<b>2</b>
<b>STRING</b>	: "Usage of sizeof
<b>Operator" The number of bytes in STRING is</b>	<b>25</b>
<b>The size of short is</b>	<b>2</b>
<b>The size of int is</b>	<b>2</b>
<b>The size of long is</b>	<b>4</b>
<b>The size of float is</b>	<b>4</b>
<b>The size of double is</b>	<b>8</b>

## **BITWISE OPERATORS:**

C supports a powerful set of operators known as bitwise operators. The bitwise operators are used for controlling and interacting with hardware and also for manipulating data at the bit level.

The Bitwise operators are as shown below table.

<b>Operator</b>	<b>Description</b>
&	bitwise AND
	bitwise OR
^	bitwise exclusive OR
<<	shift left
>>	shift right
~	one's complement

- (i) Bitwise operators cannot be applied to float or double. They can be applied to integers only.
- (ii) All except ~ operator are used in binary operations. Tilde (~) is a unary operator.
- (iii) Bitwise operators perform bitwise logical AND, bitwise logical OR, bitwise exclusive OR, bitwise shift left and shift right.
- (iv) Bitwise operators work on their operands bit by bit starting from the least significant bit. The truth table for logical bitwise operations is as shown in below table. True is taken to be 1 and false as 0.

Operand1 <b>x</b>	Operand2 <b>y</b>	$x \& y$	$x   y$	$x ^ y$	
1	1	1	1	0	
1	0	0	1	1	
0	1	0	1	1	
0	0	0	0	0	

### Bitwise AND Operator:

The Bitwise AND operator is represented by the symbol **& (ampersand)**. The result of AND operation is 1 if both operand bits are 1, otherwise, the result is zero. Further, the AND operation is done on the binary representation of the operands.

$$\begin{array}{l}
 4 \& 5 \text{ results in } 4 \\
 \text{i.e., binary value of } 4 \text{ is} \quad 0100 \\
 \text{binary value of } 5 \text{ is} \quad 0101 \\
 \hline
 \text{After AND operation} \quad 0100 \quad \text{i.e., } 4
 \end{array}$$

Bitwise **&** operator is used to check whether a particular bit is 0 or 1. Consider the number 21. Its bit pattern is represented by

0001 0101

To check whether the fourth bit is 0 or 1, AND operation must be done with another operand whose value must be  $2^4$  i.e., 16, which can be represented as 00010000.

Therefore,

$$\begin{array}{r}
 00010101 \\
 0001\ 0000 \& \text{mask} \\
 \hline
 0001\ 0000
 \end{array}$$

result shows that 4<sup>th</sup> bit is 1 since the resultant value is 16.

### Bitwise OR Operator

The bitwise OR Operator is represented by the symbol **| (vertical bar)**. It is also known as inclusive OR. The result of a bitwise OR operation is 1 if either or both corresponding bits has a value 1. Otherwise it is zero.

$$\begin{array}{l}
 \text{Consider } x = 4, y = 9 \\
 \text{x is} \quad 00000100 \quad 4 \\
 \text{y is} \quad 00001001 \quad 9 \\
 \hline
 \text{x|y is} \quad 00001101 \quad \text{i.e., } 13
 \end{array}$$

The bitwise OR is used to set a particular bit on (set to 1).

## Bitwise Exclusive OR Operator

This operator is represented by the symbol  $\wedge$  and is known as Exclusive OR operator. The result of a bitwise XOR operation is 1 if one of the corresponding bits is 1, otherwise it is 0.

**Consider the values  $x = 4, y = 9$ .**

$x$ is	00000100	4
$y$ is	00001001	9
$x \wedge y$ is	00001101	i.e., 13

## Bitwise Complement Operator

The complement operator is represented by  $\sim$  (tilde) and is also known as one's complement operator. It inverts all the bits. Thus all the zeros become 1 and all the ones become 0.

**Consider  $x = 4$**

$x$ is	00000100
$\sim x$ is	11111011

## Bitwise Shift Operator

The shift operators are used to push the bit patterns to the right or left by the number of bits given by their right operand

The **Right shift operator** is of the form :

expression $>>n$

where n is the number of bit positions to be shifted. However the rightmost n bits are lost. Further the left most n bits which are vacant will be filled with zero. For example  $x >> 4$  shifts all bits by 4 places to the right.

The **Left shift operator** is of the form

**expression $<<n$**

where n is the number of bit positions to be shifted. The left most n bits are lost and the rightmost n bits are vacated and filled with zeros.

Shift operators are used for multiplication and division by 2. For example, the statement

**$x = y << 2;$**

shifts two bits to the left in y. Further the value of x is equal to y multiplied by  $2^2$ .

Similarly, the statement

**$x = y >> 2;$**

shifts the bits in y to the right by 2. Also the value of x is equal to y divided by  $2^2$ .

## PRECEDENCE OF ARITHMETIC OPERATORS:

Any arithmetic expression is evaluated according to the rule of precedence of operators. For example in the expression.

**4\*5/2**

Evaluation takes place from left to right i.e.,

First 4 is multiplied by 5 and the result 20 is divided by 2.

The order in which the expression is evaluated depends on the hierarchy of operators

Precedence level	Operators	Associativity
1	( )	Left to right
2	unary	Right to left
3	* , / , %	Left to right
4	+ , -	Left to right

- In an arithmetic expression, the expression enclosed in parentheses is evaluated according to the precedence.
- In the case of nested parentheses, the inner most parentheses is evaluated first and the evaluation continues outwards.
- When two operators with the same precedence are encountered, the evaluation is done from left to right as in the above example. This is known as **associativity**.
- Parentheses can be used to change the order of evaluation

## PRECEDENCE AND ASSOCIATIVITY OF OPERATORS

The precedence of operators comes into effect whenever there are more than one operators involved in the expression to be evaluated.

- There are many levels of precedence, each containing one or more operators.
- The operator with the highest precedence level is evaluated first.
- Operators in the same level are evaluated either from left to right or right to left depending on their associativity. Associativity determines the operations to be performed from right and from left, when several operators of the same priority level exist in the same expression.
- The order of precedence and associativity of operators are two very important aspects of evaluation. For example, consider the statement

**(a < 10 && a + b > 15)**

where  $a = 20$ ,  $b = 1$ .

The order of evaluation will be as follows.

- (i) Since + operator has the highest priority in the expression, addition is computed first:

**$a < 10 \&& 21 > 15$**

- (ii) Both the relational operators have higher priority compared to the logical operator **&&**. Here the associativity of relational operators has to be considered which is leftto right. Thus:

**$20 < 10 \&& 21 > 15$**

**False && True**

- (iii) The result of logical **&&** is false or 0.

A complete operator precedence summary is listed in Fig. 2.20, from highest to lowest.

Operator	Operator category	Associativity	Rank
0 [] □□ .	Function call/Innermost parenthesis Array element member selection	L to R	1
+ - ++ — ! ~ * & sizeof (type)	unary plus unary minus increment decrement logicalNot One's complement Pointer reference Addressof size of a variable cast operator	R to L	2
* , /, %	Multiplication, division and remainder	L to R	3
+,-	Addition and subtraction	L to R	4
<<, >>	Left shift and Right shift	L to R	5
<, <=, >, >=	Relational operators	L to R	6
==, !=	Equality operators	L to R	7
&	Bitwise AND	L to R	8
^	Bitwise XOR (Exclusive OR)	L to R	9
	Bitwise OR (Inclusive OR)	L to R	10
&&	Logical AND	L to R	11

	Logical OR	L to R	12
? :	Conditional operator	R to L	13
=, *!=, /=, %!=, += -= &=, ^=, !=, <<=, >>=	Assignment operators and Abbreviated assignment operators	R to L	14
,	Comma operator	L to R	15

**Summary exercise:**

1. Difference between algorithm and flowchart
2. The importance of C language.
3. Different types of datatypes
4. Different types of operators in C language and its working.

**Review and revision:**

**Review and revision exercise (1) :**

**Lecture topic quiz / MCQs:**

1. The smallest unit of data in computer is \_\_\_\_\_  
a) Byte b) Nibble c) Bit d) KB ----

Answer: c

Explanation: A bit is defined as the smallest unit of data in a computer system. It is used as a short form of Binary Digit. A bit can have only two values 0 or 1. A nibble comprises of 4 bits; a byte is a collection of 8 bits whereas KB (Kilobyte) is equal to 1024 bytes.

2. Which of the following is incorrect?

Algorithms can be represented:

- |                    |                  |
|--------------------|------------------|
| a) as pseudo codes | b) as syntax     |
| c) as programs     | d) as flowcharts |

Answer: b

3. What do you call a specific instruction designed to do a task?

- |            |            |         |                |
|------------|------------|---------|----------------|
| a) Command | b) Process | c) Task | d) Instruction |
|------------|------------|---------|----------------|

Answer: a

4. An Identifier may contain?

- A) Letters a-z, A-Z in Basic character set.
- B) Underscore \_ symbol
- C) Numbers 0 to 9
- D) All the above

Answer: D

Explanation: an identifier contains letters, numbers and \_ symbol.

5. Left shift (<<) and Right shift (>>) operators are equivalent to \_\_\_\_\_ and \_\_\_\_\_ by 2.

- A) Multiplication and Division
- B) Division and Multiplication
- C) Multiplication and Remainder
- D) Remainder and Multiplication

Answer: A

Explanation: Left shift by 1 return the multiplication by 2 and Right shift by 1 return the division by 2.

#### **Review and revision exercise (2) :**

##### **Discussion questions/ Expected Questions:**

1. Draw and explain the block diagram of computer.
2. Write detailed notes on C data types
3. Write about conditional operator (ternary operator).

**Course Code: 20CS C01**

**Course Title: PROGRAMMING FOR PROBLEM SOLVING**

**UNIT-II**

**Scope:**

The Module covers the following topics: Selective, looping and nested statements. While the topics are taught using a C language, the intent of the course is to teach a programming methodology and also a laboratory component that involves development and testing of iterative and procedural programs using functions.

**Text and Reference:**

**(a) Text Book:**

Pradeep Dey and Manas Ghosh, "Programming in C", Oxford Press, 2<sup>nd</sup> Edition, 2017

**(b) Reference Books:**

E. Balaguruswamy, Programming in ANSI C, Tata McGraw-Hill.

<b>Module/ Lecture</b>	<b>Theme</b>	<b>Learning Objectives</b>
L.9	Introduction to decision control statements like If, If-else.	To learn what are various control structures, classification, if, if else, nested if else

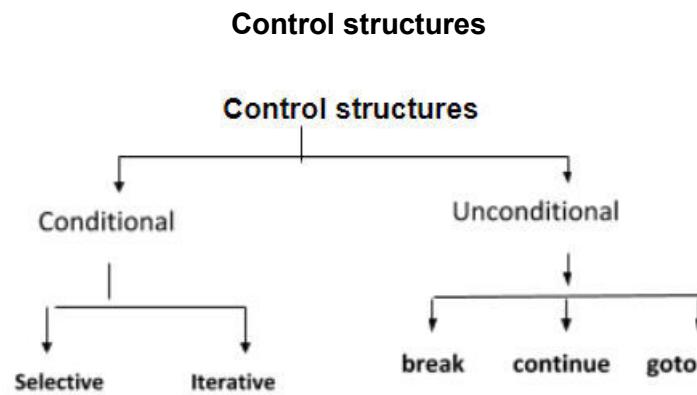
L.10	Switch-Statement, Nested Statements and Examples.	To understand else-if ladder, switch statement, use for different situations
L.11	Loop ControlStatements: For, While, Do-While and Examples.	To understand Loop statements: while, do while and for statements, how to use with the combination of other statements
L.12	Continue, Break and Go To statements.	To learn Jumping/ unconditional statements-break,continue,goto, to Use these with combination of other statements, Understand nested loops.
L.13	Case Study on control structures.	The general purpose of a case study is to: → describe an individual situation (case), in detail; → identify the key issues of the case; → analyse the case using relevant theoretical concepts from the module; → recommend a course of action for that particular case.

## Lecture -9

Objectives 1.What are various control structures

2. Classification

3. if, if else, nested if else



### Selective control statements:

1. if
2. if else
3. nested if else
4. else if ladder
5. switch

### Iterative control statements :

1. for loop,
2. while loop,
3. do while loop

### Simple if statement

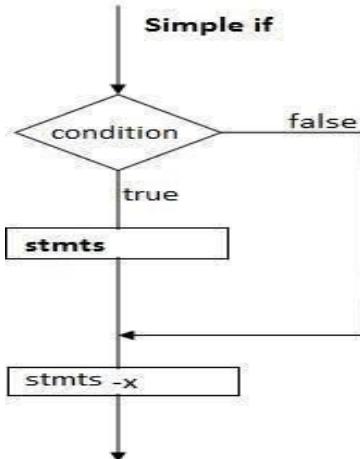
1. Simple if statement is used to make a decision “yes or no”, based on the available choice. It has the following form:

#### Syntax:

```
if ( condition )
{
stmt block;
}
stmt-x;
```

In this syntax,

- if is the keyword.
- <condition> is a relational expression or logical expression or any expression that returns either true or false. It is important to note that the condition should be enclosed within parentheses and no terminator (;) at the end.
- The *stmt block* can be a simple statement or a compound statement or a null statement.
- *stmt-x* is any valid C statement.



2. flow chart for simple if.

Whenever simple if statement is encountered, first the condition is tested. It returns either true or false. If the condition is false, the control transfers directly to stmt-x without considering the stmt block. If the condition is true, the control enters into the stmt block. Once, the end of stmt block is reached, the control transfers to stmt-x

3. **Example Program for if statement:**

```

#include<stdio.h>
main()
{
    int number;
    printf("enter a number\n");
    scanf("%d",&number);
    if(number%2==0)
        printf("Even number\n");
}

```

**Output:**

```

enter a number
12
Even number

```

**if - else statement**

if...else statement is used to make a decision based on two choices. It has the following form:

1. **Syntax:**

```

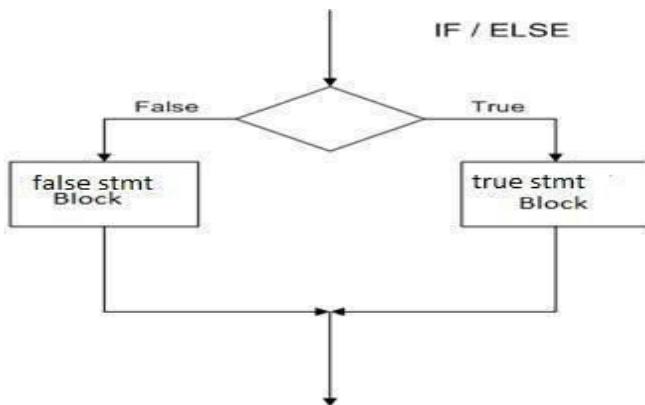
if(condition)
{
    true stmt block;
}
else
{
    false stmt block;
}
Stmt-x;

```

In this syntax,

- if and else are the keywords.
- <condition> is a relational expression or logical expression or any expression that returns either true or false. It is important to note that the condition should be enclosed within parentheses .
- The *true stmt block* and *falsestmtblock* are simple statements or compound statements or null statements.
- *stmt-x* is any valid C statement.

## 2. Flow chart:



Whenever if...else statement is encountered, first the condition is tested.  
It returns either true or false.

If the condition is true, the control enters into the true stmt block.

Once, the end of true stmt block is reached, the control transfers to stmt-x without considering else-body.

If the condition is false, the control enters into the false stmt block by skipping true stmt block.

Once, the end of false stmt block is reached, the control transfers to stmt-x.

## 3. Program for if else statement:

*/\* odd or even \*/*

```

#include <stdio.h>
main()
{
    int x;
    printf("Enter a number\n");
    scanf("%d",&x);
    if(x%2==0)
        printf("EVEN\n");
    else
        printf("ODD\n");
}
  
```

**Output:** enter a number

11  
ODD

### More examples on *if else*

**/\* largest of two numbers \*/**

```
#include <stdio.h>
main()
{
    Int x,y,big;

    if(x>y)
        big=x;
    else
        big=y;
    printf("BIG=%d\n",big);
}
```

**/\* absolute value \*/**

```
#include <stdio.h>
main()
{
    int x;
    printf("Enter a integer\n");
    scanf("%d",&x);
    if(x<0)
        printf("absolute value is %d\n",-x);
    else
        printf("absolute value is %d\n",x);
}
```

**/\* quadrant position \*/**

```
#include <stdio.h>
main()
{
    Int x,y;
    printf("Enter the coordinate\n");
    scanf("%d%d",&x,&y);
    if(x>0 && y>0)
        printf("FIRST QUADRANT\n");

    if(x<0 && y>0)
        printf("SECOND QUADRANT\n");

    if(x<0 && y<0)
        printf("THIRD QUADRANT\n");

    if(x>0 && y<0)
```

```

printf("FOURTH QUADRANT\n");

}

/* alphabet or not */
#include <stdio.h>
main()
{
char ch;
printf("Enter a character\n");
scanf("%c",&ch);
if(ch>= 'a' &&ch<='z')
    printf("alphabet\n");
else
    printf("not a alphabet\n");
}

/* vowel or not */
#include <stdio.h>
main()
{
char ch;
printf("Enter a alphabet\n");
scanf("%c",&ch);
if(ch== 'a' || ch=='e'||ch=='i'||ch=='o'||ch=='u')
printf("vowel\n");
else
printf("not a vowel\n");
}

```

### **Nested if—else statement**

Nested if...else statement is one of the conditional control-flow statements. If the body of if statement contains at least one if statement, then that if statement is called as —Nested if...else statement|. The nested if...else statement can be used in such a situation where at least two conditions should be satisfied in order to execute particular set of instructions. It can also be used to make a decision among multiple choices. The nested if...else statement has the following form:

**Syntax:** if(condition1)  
{  
 if(condition 2)  
 {  
 stmt1;  
 }  
 else  
 {  
 stmt2;  
 }  
}

```

}
else
{
    stmt 3;
}

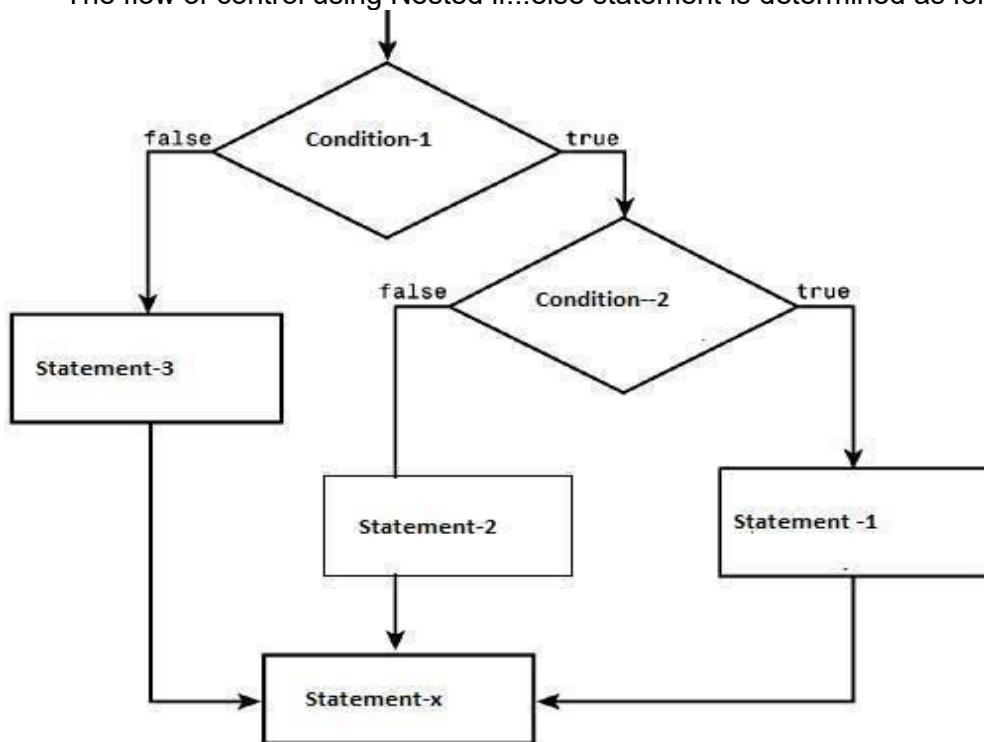
stmt-x;

```

In this syntax,

- if and else are keywords.
- <condition1>, <condition2> ... <condition> are relational expressions or logical expressions or any other expressions that return true or false. It is important to note that the condition should be enclosed within parentheses ( and ).
- if-body and else-body are simple statements or compound statements or empty statements.
- stmt-x is a valid C statement.

The flow of control using Nested if...else statement is determined as follows



Whenever nested if...else statement is encountered, first <condition1> is tested. It returns either true or false.

If condition1 (or outer condition) is false, then the control transfers to else-body (if exists) by skipping if-body.

If condition1 (or outer condition) is true, then condition2 (or inner condition) is tested. If the condition2 is true, if-body gets executed. Otherwise, the else-body that is inside of if statement gets executed.

### **Program for nested if... else statement:**

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

    printf("enter three values\n"); scanf("%d%d%d",&a,&b,&c); if(a>b)
    {
        if(a>c)
            printf("%d\n",a);
        else
            printf("%d\n",b);
    }

    else
    {
        if(c>b)
            printf("%d\n",c);
        else
            printf("%d\n",b);
    }
}
```

**Output:** enter three values 3 5 12  
12

### **Programming Problems:**

1. Develop a C program to check whether a number is negative, positive or zero.(L3) (CO3)
2. Develop a C program to check whether a number is divisible by 5 and 11 or not.(L3) (CO3)
3. Develop a C program to check whether a year is leap year or not.(L3) (CO3)
4. Develop a C program to input any alphabet and check whether it is vowel or consonant.(L3) (CO3)
5. Develop a C program to input any character and check whether it is alphabet, digit or special character.(L3) (CO3)
6. Develop a C program to check whether a character is uppercase or lowercase alphabet.(L3) (CO3)
7. Develop a C program to input angles of a triangle and check whether triangle is valid or not.(L3) (CO3)
8. Develop a C program to input all sides of a triangle and check whether triangle is valid or not.(L3) (CO3)
9. Develop a C program to check whether the triangle is equilateral, isosceles or scalene triangle.(L3) (CO3)

10. Develop a C program to find all roots of a quadratic equation.(L3) (CO3)

## Lecture- 10

Objectives:

1. To understand else-if ladder
2. To understand switch statement
3. Learn to use for different situations

### if-else if STATEMENT OR else if ladder

Introduction

If we want to make one decisions out of many options, then we can use the if-else if statement.

Syntax

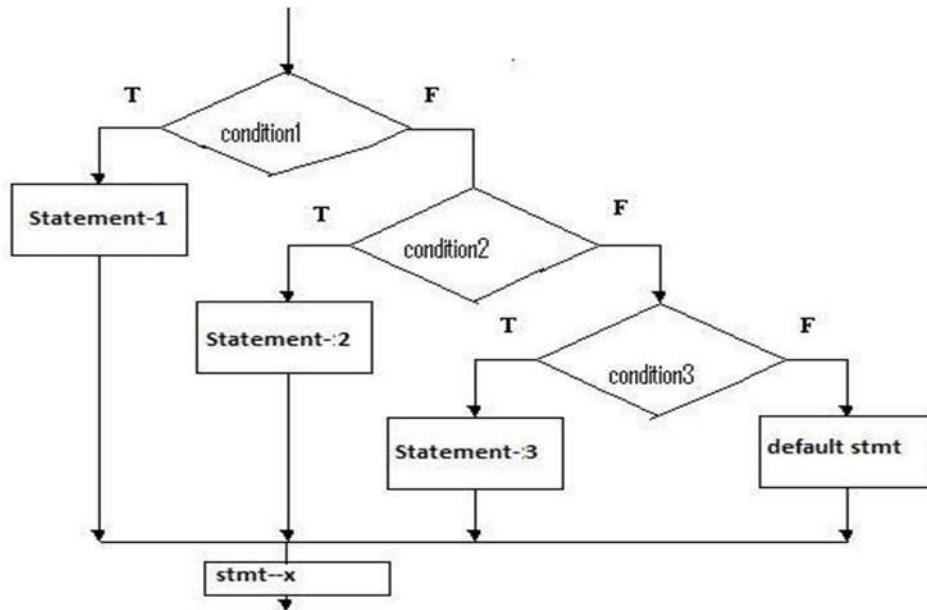
The general format for the if-else if statement is:

```
if (condition 1)
    simple or compound statement // s1
else if (condition 2)
    simple or compound statement // s2
else if ( condition 3)
    simple or compound statement // s3
.....
else if ( conditon n )
    simple or compound statement // sn
```

If condition 1 is true then s1 is executed.

If condition 1 is false and condition 2 is true then s2 is executed.

The else clause is always associated with the nearest unresolved if statement.



Whenever else if ladder is encountered, condition1 is tested first. If it is true, the statement 1 gets executed. After then the control transfers to stmt-x.

If condition1 is false, then condition2 is tested. If condition2 is false, the other conditions are tested. If all are false, the default stmt at the end gets executed. After then the control transfers to stmt-x.

If any one of all conditions is true, then the body associated with it gets executed. After that the control transfers to stmt-x.

#### Points to Remember

1. We can use if-else if when you want to check several conditions but still execute one statement.
2. When writing an if-else if statement, be careful to associate else statement to the appropriate if statement.
3. We must have parentheses around the condition.

#### Example program:

```

/* to find grade when percentage of marks are given */
#include<stdio.h>
void main()
{
    int m1,m2,m3,avg,tot;
    printf("enter three subject marks");
    scanf("%d%d%d",&m1,&m2,&m);
    tot=m1+m2+m3;
    avg=tot/3;
    if(avg>=75)
    {
        printf("distinction");
    }
}
  
```

```

else if(avg>=60 &&avg<75)
{
    printf("first class");
}
else if(avg>=50 &&avg<60)
{
    printf("second class");
}
else if (avg<50)
{
    printf("fail");
}

```

Output: enter three subject marks  
 85 80 81  
 Distinction

## THE switch STATEMENT

### Introduction

We can use a switch statement when we want to check multiple conditions. It can also be done using an if statement but it will be too lengthy and difficult to debug.

### Syntax

The general format for a switch statement is

```
switch(integer expression)
```

```

{
case 1:
    statement;
break;
case 2:
    statement;
break;
-----
default:
    statement;
}
```

Example of a case constant expression and column:

```

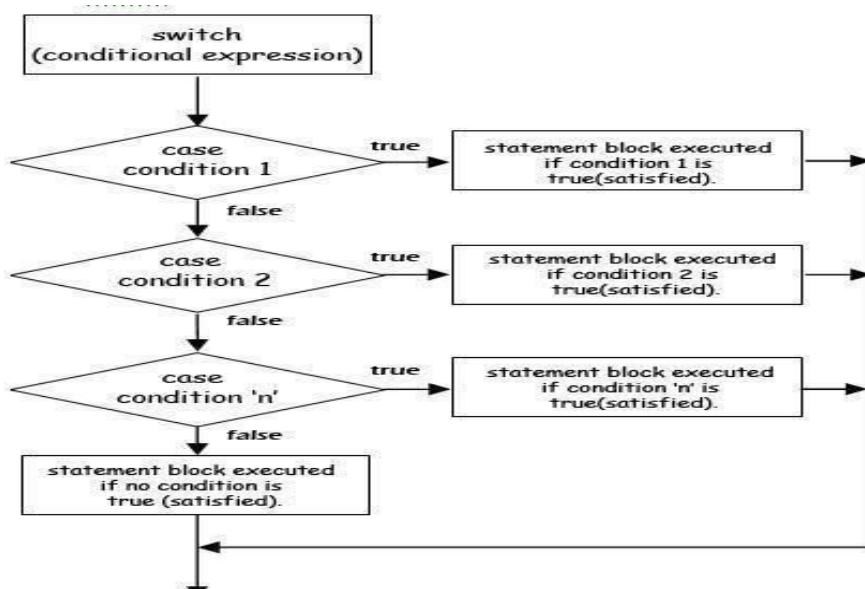
switch (i/10)
{
    case 0: printf ("Number less than 10"); // A
    break;
    case 1: printf ("Number less than 20"); // B
    break;
    case 2: printf ("Number less than 30"); // C
    break;
    default: printf ("Number greater than or equal to 40"); // D
    break;
}
```

### Explanation

1. The switch expression should be an integer expression and, when evaluated, it must have an integer value.

2. The case constant expression must represent a particular integer value and no two case expressions should have the same value.
3. The value of the switch expression is compared with the case constant expression in the order specified, that is, from the top down.
4. The execution begins from the case where the switch expression is matched and it flows downward.
5. In the absence of a break statement, all statements that are followed by matched cases are executed.
6. If there is no matched case then the default is executed. You can have either zero or one default statement.
7. In the case of a nested switch statement, the break statements break the inner switch statement.

Flow chart:



Whenever, switch statement is encountered, first the value of <exp> gets matched with case values. If suitable match is found, the statements block related to that matched case gets executed. The break statement at the end transfers the control to the Next-statement.

If suitable match is not found, the default statements block gets executed and then the control gets transferred to Next-statement.

#### Point to Remember

1. case constructs need not be serial.
2. characters can be used.
3. combination of integers and characters is possible.
4. default is optional.
5. Brace in case block is optional

```
//switch 1
// to display name of the day
#include <stdio.h>
void main()
{
    int day;
    printf("Enter day number\n");
    scanf("%d",&day);
    switch(day)
    {
        case 1:
            printf("monday\n");
            break;
        case 2:
            printf("tuesday\n");
            break;
        case 3:
            printf("wednesday\n");
            break;
        case 4:
            printf("thursday\n");
            break;
        case 5:
            printf("friday\n");
            break;
        case 6:
            printf("saturday\n");
            break;
        case 7:
            printf("sunday\n");
            break;
        default:
            printf("Wrong entry\n");
    }
}
//switch 2
// to display the gender
#include <stdio.h>
void main()
{
    char ch;
    printf("Enter your gender\n");
    scanf("%c",&ch);
    switch(ch)
```

```

{
    case 'm':
    case 'M':
        printf("so you are a male\n");
        break;

    case 'f':
    case 'F':
        printf("so you are a female\n ")
        break;

    default:
        printf("Wrong entry\n");
}
}

```

**Program exercises:**

1. Develop a C program to input week number and print week day.(L3)(CO3)
2. Develop a C program to input month number and print number of days in that month.(L3)(CO3)
3. Develop a C program to input marks of five subjects Physics, Chemistry, Biology, Mathematics and Computer. Calculate percentage and grade according to following:(L3)(CO3)
  - Percentage >= 90% : Grade A
  - Percentage >= 80% : Grade B
  - Percentage >= 70% : Grade C
  - Percentage >= 60% : Grade D
  - Percentage >= 40% : Grade E
  - Percentage < 40% : Grade F
4. Develop a C program to input basic salary of an employee and calculate its Gross salary according to following:(L3)(CO3)
  - Basic Salary <= 10000 : HRA = 20%, DA = 80%
  - Basic Salary <= 20000 : HRA = 25%, DA = 90%
  - Basic Salary >20000 : HRA = 30%, DA = 95%
5. Develop a C program to input electricity unit charges and calculate total electricity bill according to the given condition:(L3)(CO3)
  - For first 50 units Rs. 0.50/unit
  - For next 100 units Rs. 0.75/unit
  - For next 100 units Rs. 1.20/unit
  - For unit above 250 Rs. 1.50/unit
 An additional surcharge of 20% is added to the bill(L3)(CO3)
6. Develop a C program print total number of days in a month using switch case.(L3)(CO3)
7. Develop a C program to check whether an alphabet is vowel or consonant using switch case.(L3)(CO3)
8. Develop a C program to find maximum between two numbers using switch case.(L3)(CO3)
9. Develop a C program to find roots of a quadratic equation using switch case.(L3)(CO3)
10. Develop a C program to create Simple Calculator using switch case.  
(L3)(CO3)

## LECTURE-11

### Objective:

1. Loop statements: while, do while and for statements
2. How to use with the combination of other statements.

### while LOOP

#### Introduction

The while loop is used when we want to repeat the execution of a certain statement or a set of statements (compound statement).

#### Syntax

The general format for a while loop is

while (condition)

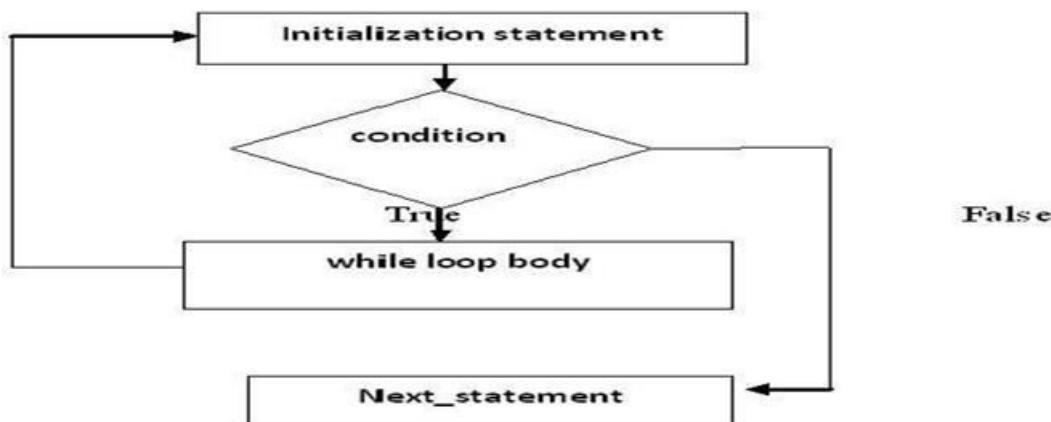
simple or compound statement (body of the loop)

For example,

```
i = 0;  
while (i<5)  
{  
    printf(" the value of i is %d\n", i);  
    i = i + 1;  
}
```

#### Explanation

1. Before entering into the loop, the while condition is evaluated. If it is true then only the loop body is executed.
2. Before making an iteration, the while condition is checked. If it is true then the loop body is executed.
3. It is the responsibility of the programmer to ensure that the condition is false after certain iterations; otherwise, the loop will make infinite iterations and it will not terminate.
4. The programmer should be aware of the final value of the looping variable. For example, in this case, the final value of the looping variable is 5.
5. While writing the loop body, you have to be careful to decide whether the loop variable is updated at the start of the body or at the end of the body.



```
#include<stdio.h>  
#include<conio.h> int main()  
{  
int i,n;  
printf("enter the range\n");  
scanf("%d",&n);
```

```

i=1;

while(i<=n)

{
printf("%d ",i); i=i+1;
}
}

```

Output: enter the range 10  
1 2 3 4 5 6 7 8 9 10

### do-while LOOP

#### Introduction

The do-while loop is used when we want to execute the loop body at least once. The do-while loop executes the loop body and then traces the condition.

#### **Syntax**

The general format for a do-while loop is  
do  
simple or compound statement  
while (condition);

For example,

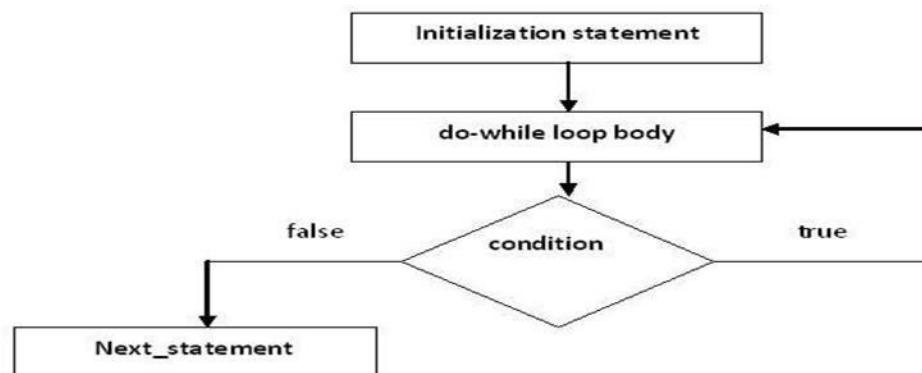
```

i = 0;
do
{
printf(" the value of i is %d\n", i);
    i = i + 1;
}
while (i<5);

```

#### **Explanation**

1. The loop body is executed at least once.
2. The condition is checked after executing the loop body once.
3. If the condition is false then the loop is terminated.
4. In this example, the last value of i is printed as 5.



```
#include<stdio.h>

#include<conio.h> int main()
{
int i,n;
printf("enter the range\n"); scanf("%d",&n);
i=1;
do
{
printf("%d\n",i);
i=i+1;
}
while(i<=n);
}
```

Output: enter the range 5

```
1
2
3
4
5
```

## for LOOP

### Introduction

The for loop is used only when the number of iterations is predetermined, for example, 10 iterations or 100 iterations.

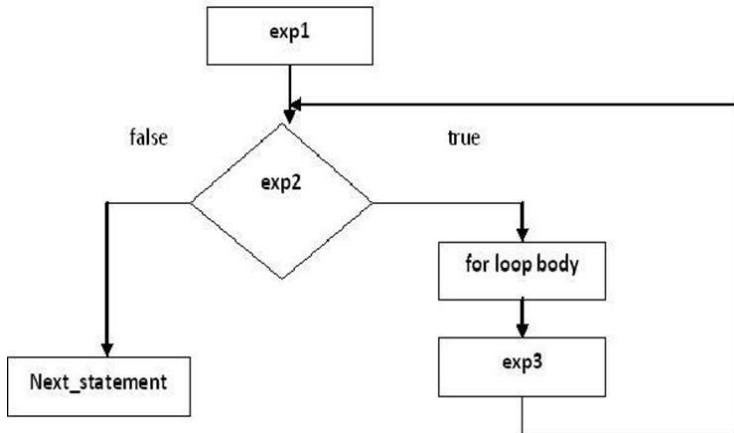
### **Syntax**

The general format for the for loop is  
**for (initializing; continuation condition; update)**  
 simple or compound statement

For example,  
**for (i = 1; i<=5; i++)**  
{  
printf("%d", i);  
}

### **Explanation**

1. The for loop has four components; three are given in parentheses and one in the loop body.
2. All three components between the parentheses are optional.
3. The initialization part is executed first and only once.
4. The condition is evaluated before the loop body is executed. If the condition is false then the loop body is not executed.
5. The update part is executed only after the loop body is executed and is generally used for updating the loop variables.
6. The absence of a condition is taken as true.
7. It is the responsibility of the programmer to make sure the condition is false after certain iterations.



Program to print n natural numbers using for loop

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

```
#include<conio.h>
void main()
{
int i,n;
printf("enter the value"); scanf("%d ",&n);
for(i=1;i<=n;i++)
{
printf("%d\n",i);
}
}
```

Output: enter the value 5

1 2 3 4 5

### THE for LOOP WITH A COMMA OPERATOR

#### Introduction

We may want to control the loop variables in the same for loop. We can use one for loop with a comma operator in such situations.

#### Syntax

```
for (i = 0, j = 10; i < 3 && j > 8; i++, j-)
printf (" the value of i and j %d %d\n",i, j);
```

#### Explanation

1. First i is initialized to 0, and j is initialized to 10.
2. The conditions  $i < 3$  and  $j > 8$  are evaluated and the result is printed only if both conditions are true.
3. After executing the loop body, i is incremented by 1 and j is decremented by 1.
4. The comma operator also returns a value. It returns the value of the rightmost operand.  
The value of  $(i = 0, j = 10)$  is 10.

### 1. Program to find the factorial of a number

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

```
main()
```

```

{
longint fact=1;
int i,n;
    printf("Enter a positive number to find factorial\n");
    scanf("%d",&n);
    for(i=1; i<n; i++)
        fact *=i;
    printf("The factorial of %d is %ld",n,fact);
}

2. Fibonacci series
main()
{
    int i, f1, f2, f3, n;
    f1 = f2 = 1;

    printf( "\nHow many Fibonacci numbers required ? ");
    scanf("%d", &n);
    printf("\nThe first %3d Fibonacci numbers are\n", n);
    printf("%5d\n", f1);
    printf("%5d\n", f2);
    for( i = 3; i <= n; i++)
    {
        f3 = f1 + f2;
        printf("%5d\n", f3);
        f1 = f2;
        f2 = f3;
    }
}

//3. Sine series
/* sin(x) = x - x^3/3! + x^5/5! + ... */
#include <stdio.h>
main()
{
int i, n;
float x,denom, term, sum,deg;
printf("Enter value of x in deg and number of
terms: \n");
scanf("%f%d", &deg, &n);
x=deg*3.14/180;
sum = term = x;
for ( i = 2; i <= n; i++)
{
    denom = (2 * i - 2)*( 2 * i - 1);
    term *= -(x*x)/ denom;
    sum += term;
}
printf("\ndeg = %.1f, Sum = %.2f", deg, sum);
} /* end of main */

/* cos(x) = 1 - x^2/2! + x^4/4! + ... */

```

### **// cosine series**

```

main()
{
int i, n;

```

```

float      x,denom, term, sum,deg;
printf("Enter value of x in deg and number of terms: \n");
scanf("%f%d", &deg, &n);
x=deg*3.14/180;
sum = term = 1;
for ( i = 1; i<= n; i++)
{
    denom = (2 * i)*( 2 * i - 1);
    term   *= -(x*x)/ denom;
    sum   += term;
}
printf("\ndeg = %.1f,    Sum = %.2f", deg, sum);
} /* end of main */

```

**// to find the sum of digits of a number**

```

#include <stdio.h>
main()
{
    int n,sum=0,digit;;
    printf("Enter a number\n");
    scanf("%d",&n);
    while(n!=0)
    {
        digit=n%10;
        sum=sum+digit;
        n=n/10;
    }
    printf("%d",sum)
}

```

**// to convert decimal number to binary number**

```

#include <stdio.h>
main()
{
    int n,sum=0,digit,i=0;
    printf("Enter a number\n");
    scanf("%d",&n);
    while(n!=0)
    {
        digit=n%2;
        sum=sum+digit*pow(10,i);
        n=n/2;
        i++;
    }
    printf("%d",sum);
}

```

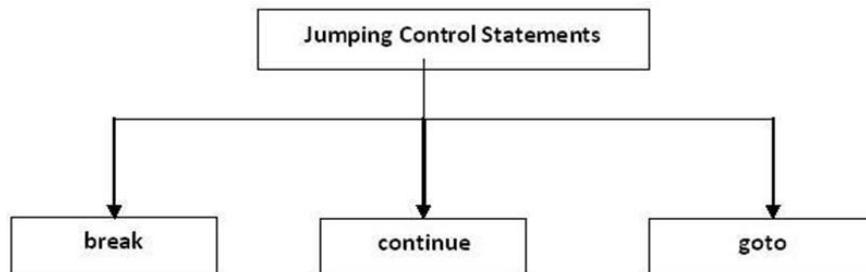
**Programming exercises:**

1. Write a program to find  $n^{\text{th}}$  Fibonacci number.
2. Write a program to generate first  $n$  terms of Fibonacci series.
3. Write a program to find factorial of a number.
4. Write a program to check whether the given number is prime or not.
5. Write a program to print first  $n$  prime numbers.
6. Write a program to print prime numbers upto  $n$ .
7. Write a program to print prime numbers between  $n_1$  and  $n_2$ .
8. WAP to display menu for selecting addition, subtraction, multiplication and division and compute according to menu chosen. (Use do-while).
9. Using while loop, write a program to print all Armstrong numbers between 1 and 500. If sum of cubes of each digit of the numbers is equal to the number itself, then number is called Armstrong number.  
For example:  $153 = (1*1*1) + (5*5*5) + (3*3*3)$
10. WAP to find whether the given number is strong or not using functions.  
(Example:  $145 = 1! + 4! + 5! = 1 + 24 + 120 = 145$  )

## Lecture- 12

Objectives:

1. Jumping/ unconditional statements- break,continue,goto
2. Use these with combination of other statements.
3. Understand nested loops.



### break statement

The break statement is used within the looping control statements, switch statement and nested loops. When it is used with the for, while or do-while statements, the control comes out of the corresponding loop and continues with the next statement.

When it is used in the nested loop or switch statement, the control comes out of that loop / switch statement within which it is used. But, it does not come out of the complete nesting.

Syntax for the —break statement is:

```
break;
```

In this syntax, break is the keyword.

The following representation shows the transfer of control when break statement is used:

Any loop

```
{
```

```
statement_1;  
statement_2;  
:  
  
break;  
:  
  
}  
  
next_statement;
```

Program for break statement:

```
#include<stdio.h>  
  
#include<conio.h>  
int main()  
{  
int i;  
  
for(i=1; i<=10; i++)  
{  
if(i==6)  
break;  
printf("%d",i);  
}  
}
```

Output: 12345

## 2. continue statement

A continue statement is used within loops to end the execution of the current iteration and proceed to the next iteration. It provides a way of skipping the remaining statements in that iteration after the continue statement. It is important to note that a continue statement should be used only in loop constructs and not in selective control statements.

Syntax for continue statement is:

```
continue;
```

where continue is the keyword.

The following representation shows the transfer of control when continue statement is used:  
Any loop

```
{  
  
statement_1; statement_2;  
:  
  
continue;
```

```
:  
}  
  
next_statement;
```

Program for continue statement:

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

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

```
int i, sum=0, n;  
for(i=1; i<=10; i++)  
{
```

```
printf("enter any no:");  
scanf("%d",&n);  
if(n<0)  
continue;  
else  
sum=sum+n;  
printf("%d\n",sum);  
}
```

```
}
```

Output: enter any no:8

18

### 3. goto statement

The goto statement transfers the control to the specified location unconditionally. There are certain situations where goto statement makes the program simpler. For example, if a deeply nested loop is to be exited earlier, goto may be used for breaking more than one loop at a time. In this case, a break statement will not serve the purpose because it only exits a single loop.

Syntax for goto statement is:

```
label:  
{  
statement_1; statement_2;  
:  
}  
:  
goto label;
```

In this syntax, goto is the keyword and label is any valid identifier and should be ended with a colon (:).

The identifier following goto is a statement label and need not be declared. The name of the statement or label can also be used as a variable name in the same program if it is declared appropriately. The compiler identifies the name as a label if it appears in a goto statement and as a variable if it appears in an expression.

If the block of statements that has label appears before the goto statement, then the control has to move to backward and that goto is called as backward goto. If the block of

statements that has label appears after the goto statement, then the control has to move to forward and that goto is called as forward goto.

Program for goto statement:

```
#include<stdio.h>

#include<conio.h> void main()
{
    printf("www."); goto x;
    y:
    printf("expert"); goto z;
    x:
    printf("c programming"); goto y;
    z:
    printf(".com");
}
```

Output: www.c programming expert.com

### **PROGRAM 1**

```
1
1 2
1 2 3
#include <stdio.h>
void main()
{
    int line ,n;
    for(line=1;line<=3;line++)
    {
        for(n=1;n<=line; n++)
            printf("%3d",n);
            printf("\n");
    }
}
```

### **PROGRAM2**

```
1
```

```

1   2
1   2   3
#include <stdio.h>
void main()
{
    int line ,n,gap;
    for(line=1;line<=3;line++)
    {
        for(gap=1;gap<=20-line;gap++)
            printf(" ");
        for (n=1;n<=line;n++)
            printf("%4d",n);
        printf("\n");
    }
}

```

### PROGRAM 3

```

/*
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```

```

*/
#include <stdio.h>
void main()
{
    int line ,n,gap;

    for(line=1;line<=5;line++)
    {
        for(gap=1;gap<=5-line;gap++)
            printf(" ");
        for(n=1;n<=line;n++)
            printf("%2d",n);
        printf("\n");
    }
}

```

### PROGRAM 4

```

1
1 2 1
1 2 3 1 2
1 2 3 4 1 2 3
1 2 3 4 5 1 2 3 4

```

```

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

    for(line=1;line<=5;line++)
    {
        for(gap=1;gap<=20-line;gap++)
            printf(" ");

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

```

```

        printf("%2d",n);

        for(n=1;n<line;n++)
            printf("%2d",n);
            printf("\n");
        }
    }

//program 5 :Pascal triangle
#include <stdio.h>
void main()
{
int binom,n,q,g,x;
binom=1;
q=0;
printf("enter no of rows\n");
scanf("%d",&n);
while(q<n)
{
for(g=40-3*q;g>0;g--)
printf(" ");
for(x=0;x<=q;x++)
{
if((x==0)||((q==0))
binom=1;
else
binom = (binom*(q-x+1))/x;
printf("%6d",binom);
}
printf("\n");
q++;
}
}

```

### **Programming exercises:**

Develop programs to display the given patterns (L3)(CO3)

```

*****
*  *
*  *
*  *
*****
```

```

*****
*****
```

\*\*\*\*\*

\*\*\*\*\*

\*\*\*\*\*

\*

\*\*

\* \*

\* \*

\*\*\*\*\*

10101  
01010  
10101  
01010  
10101

5 5 5 5 5 5 5 5  
5 4 4 4 4 4 4 5  
5 4 3 3 3 3 3 4 5  
5 4 3 2 2 2 3 4 5  
5 4 3 2 1 2 3 4 5  
5 4 3 2 2 2 3 4 5  
5 4 3 3 3 3 3 4 5  
5 4 4 4 4 4 4 4 5  
5 5 5 5 5 5 5 5

## Lecture-13

### Case Study on control structures:

1. To display the given decimal number in words
2. To display the given decimal number in roman numbers
3. To find the day name when date is entered.

```

// to display the given number in words
#include <stdio.h>
void main()
{
    intnum,digit,temp,rev=0;
    printf("Enter a number\n");
    scanf("%d",&num);
    temp=num;
    while(num!=0)
    {
        digit=num%10;
        rev = rev*10 + digit;
        num=num/10;
    }
    while(rev!=0)
    {
        digit=rev%10;
        switch(digit)
        {
            case 1:
                printf("ONE ");
                break;
            case 2:
                printf("TWO ");
                break;
            case 3:
                printf("THREE ");
                break;
            case 4:
                printf("FOUR ");
                break;
            case 5:
                printf("FIVE ");
                break;
            case 6:
                printf("SIX ");
                break;
            case 7:
                printf("SEVEN ");
                break;
            case 8:
                printf("EIGHT ");
                break;
            case 9:
                printf("NINE ");
                break;
            case 0:
                printf("ZERO ");
        }
    }
}

```

```

        }
        rev=rev/10;
    }

while(temp%10==0)
{
    printf("ZERO ");
    temp=temp/10;
}

```

## Short answer questions

1. What are the control structures available in C(L1)(CO3)  
If, if else, nested if else, switch, for loop, while loop, do while loop, break, continue and goto
2. Name the selective constructs in c(L1)(CO3)  
If,ifelse,nested if else, else if ladder, switch
3. Distinguish between a pre test looping and post test looping(L4)(CO3)
4. Name the loop constructs in c. (L1)(CO3)
5. Name the unconditional control statements in C(L1)(CO3)
6. Name the conditional control structures in C. (L1)(CO3)
7. Explain in detail the different forms of **if** constructs in c. (L2)(CO3)
8. What is conditional expression(L1)(CO3)
9. Illustrate the use of ternary operations. (L2)(CO3)
10. Explain the **while** structure with examples(L2)(CO3)
11. Explain the **do-while** structure with examples(L2)(CO3)
12. Can the action of a **do-while** structure is simulated by **if** or **if-else** structures?  
(L4)(CO3)
13. Explain with an example.(L2)(CO3)
14. Compare the **while** structure with **do-while** structure
15. Write the advantages of **while** loop over the **do-while loop**(L2)(CO3)
16. How is a for loop equivalently written using a while loop.(L2)(CO3)
17. Under what circumstances will the do-while be more appropriate than the for loop.(L2)(CO3)
18. What is a comma expression?(L2)(CO3)
19. What is the purpose of continue statement?(L2)(CO3)
20. What is the propose of break statement?(L2)(CO3)
21. Differentiate between break and continue statements.(L4)(CO3)
22. What is meant by nested loop?(L2)(CO3)
23. Explain switch-case structure with example.(L2)(CO3)
24. Compare the switch-case with if else structure.(L4)(CO3)
25. Why goto statements are discouraged?(L2)(CO3)
26. Rewrite the following without using compound relations(L2)(CO3)
 

```

if(EM>90&&EG>95||Total>190)
    printf("SUCCESS");
else
    printf("Failure");

```
26. what is the output of the following program(L2)(CO3)
 

```

main()
{

```

```

int i;
for(;i;)
i--;
printf("%d",i);
}

```

27. Distinguish between source code, object code and executable code.(L4)(CO3)
28. Draw flow chart for various loops.(L3)(CO3)
29. Draw flow chart for various control constructs.(L3)(CO3)

**Programs:**

1. Develop a program to find sum of digits of a given integer number.(L3)(CO3)
2. Develop a program to print the given integer number in reverse order.(L3)(CO3)
3. Develop a program to check whether the given number is palindrome or not.(L3)(CO3)
4. Develop a program to find largest of given two numbers.(L3)(CO3)
5. Develop a program find largest of given three numbers by using simple if, nested if, else if ladder(L3)(CO3)
6. Develop a program to check whether the given triangle is right triangle or not(L3)(CO3)
7. Develop a program to find the quadrant position of given coordinate.(L3)(CO3)
8. Develop a program to check whether the entered character is vowel or not.(L3)(CO3)
9. Develop a program to print CBIT twenty times by using for loop.(L3)(CO3)
10. Develop a program to find the sum of first n natural numbers.(L3)(CO3)
11. Develop a program to find the sum of first n even numbers.(L3)(CO3)
12. Develop a program to find the sum of given n numbers.(L3)(CO3)
13. Develop a program to print n<sup>th</sup> table upto 10.(L3)(CO3)
14. Develop a program to print the following triangle of numbers by using nested loops.(L3)(CO3)

```

1
1 2
1 2 3
1 2 3 4

```

```

1
1 2
1 2 3
1 2
1

```

```

1
1 2
1 2 3
1
1 2 1
1 2 3 2 1
1

```

2 1 2  
3 2 1 2 3

- 15.** Write a program to print the sum of first n terms of sine series.(L3)(CO3)
- 16.** Write a program to print the sum of first n terms of cosine series.(L3)(CO3)
- 17.** Write a program to print the Pascal triangle.(L6)(CO3)
- 18.** Write a program to find  $n^{\text{th}}$  Fibonacci number.(L3)(CO3)
- 19.** Write a program to generate first n terms of Fibonacci series.(L3)(CO3)
- 20.** Write a program to find factorial of a number.(L3)(CO3)
- 21.** Write a program to check whether the given number is prime or not.(L3)(CO3)
- 22.** Write a program to print first n prime numbers.(L3)(CO3)
- 23.** Write a program to print prime numbers upto n.(L3)(CO3)
- 24.** Write a program to print prime numbers between n1 and n2.(L3)(CO3)
- 25.** WAP to display menu for selecting addition, subtraction, multiplication and division and compute according to menu chosen. (Use do-while).(L3)(CO3)
- 26.** WAP to addition, subtraction, and multiplication of two complex numbers  $(a+ib)$  and  $(x+iy)$ .(L3)(CO3)
- 27.** Using while loop, write a program to print all Armstrong numbers between 1 and 500. If sum of cubes of each digit of the numbers is equal to the number itself, then number is called Armstrong number.(L3)(CO3)  
For example:  $153 = (1*1*1) + (5*5*5) + (3*3*3)$
- 28.** WAP to find whether the given number is strong or not using functions.(L3)(CO3)  
(Example:  $145 = 1! + 4! + 5! = 1 + 24 + 120 = 145$  )
- 29.** Input a year from keyboard and determine whether it is leap year or not using the conditional operator.(L3)(CO3)
- 30.** Accept three numbers from user and determine which of the two numbers or all three numbers are equal or not using logical operators.(L3)(CO3)
- 31.** WAP to calculate  $nPr = n!/(n-r)!$ (L3)(CO3)
- 32.** WAP to calculate  $nCr = n! / n!*(n-r)!$ (L3)(CO3)
- 33.** WAP to find whether the given number is perfect or not  
Example ( 6 is perfect number since  $6 = 1 + 2 + 3$  )(L3)(CO3)
- 34.** WAP to convert given decimal number to binary number by using a function.(L3)(CO3)



# CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. [www.cbit.ac.in](http://www.cbit.ac.in)



COMMITTED TO  
RESEARCH,  
INNOVATION AND  
EDUCATION

**41**  
years

**Course Code:** 20CS C01

**Course Title:** PROGRAMMING FOR PROBLEM SOLVING

**Instructors Name:**

Name	Email
Smt. Venkata Sushma Chinta	venkatasushmachinta_mech@cbit.ac.in

**Objective:**

The primary goals of the module functions are to introduce:

- The major objective is to provide students with understanding of code organization and functional decomposition with using different data types.

**Outcome:**

On Successful completion of the module: functions, students will be able to:

- Decompose a problem into modules and use functions to implement the modules.

**Scope:**

The Module covers the following topics: Basics of Functions, User-defined Functions, Inter Function Communication, Standard Functions, Storage Classes (Auto, Register, Static, Extern), Scope Rules. While the topics are taught using a C language, the intent of the course is to teach a programming methodology and also a laboratory component that involves development and testing of iterative and procedural programs using functions.

**Text and Reference:****(a) Text Book:**

Pradeep Dey and Manas Ghosh, "Programming in C", Oxford Press, 2<sup>nd</sup> Edition, 2017

**(b) Reference Books:**

E. Balaguruswamy, Programming in ANSI C, Tata McGraw-Hill.

<b>Module/ Lecture</b>	<b>Theme</b>	<b>Learning Objectives</b>
L.14	Basics of Functions, User-defined Functions.	To learn basic representation of functions, and the way they are declared.
L.15	Inter Function Communication, Standard Functions.	To learn basic types of standard functions and recursive functions
L.16	Parameter Passing-Call-by-value, call-by-reference	To learn Parameter Passing-Call-by-value, call-by-reference
L.17	Storage Classes (Auto, Register, Static, Extern), Scope Rules.	To learn Storage Classes (Auto, Register, Static, Extern), Scope Rules.
L.18	Case Study on Functions	The general purpose of a case study is to: → describe an individual situation (case), in detail; → identify the key issues of the case; → analyse the case using relevant theoretical concepts from the module; → recommend a course of action for that particular case.

Lecture title	L.14 to L.18
Date	

**Key topics:** Introduction, uses of functions, Function definition, declaration, passing parameters to functions, recursion, scope of variables and storage classes.

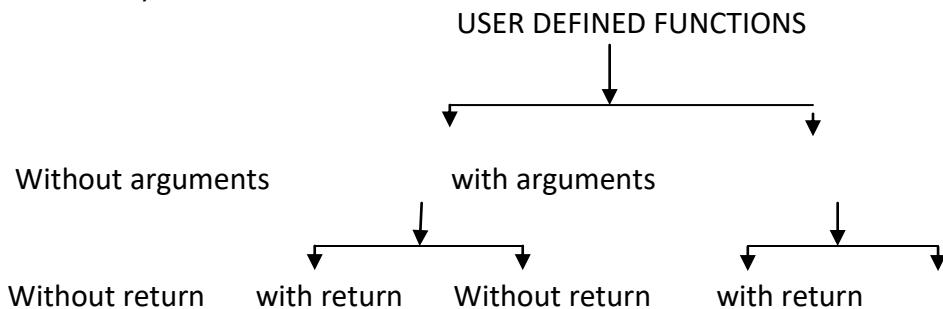
Case study using functions.

#### **Definition:**

A function is a set of instructions under a name that carries out a specific task, assigned to it.

#### **CLASSIFICATION of functions:**

1. User defined functions (UDF)
2. Library functions



#### **Example:**

**void sum( )**                  function without arguments and without return value  
**int sum( )**                  function without arguments but with return value  
**void sum(int ,int)**          function with arguments but without return value  
**int sum(int,int)**          function with arguments and with return value

1. It avoids the need for redundant programming of the same instructions.
2. Logical clarity of the programming will be clear.
3. Easy to debug.
4. A library with user defined functions can be created.

#### **Function prototype:**

If we use any variable in the program, we declare that variable in the declaration section of the program. Similarly, any function that is to be called should be declared before its call in the calling function. This ***declaration of a function is called function prototype.***

Function declaration should be done either in the declaration section of calling function or before it. The function prototype should agree with definition of that function.

Prototype contains the data type returned by the function followed by the name of that function and a pair of parentheses enclosing the data type of each parameter separately.

Optionally the name of each parameter may be given.

Ex: int fact(int); or int fact(int n);

If there are no parameters, void should be shown in parentheses. Similarly, if there is no return value, return value type should be indicated as void.

### ***Function definition***

Writing code for the function is called function definition.

Function definition contains function header + body of function.

Syntax of function header :*return type      name of the function ( arguments)*

### ***FUNCTION CALL***

Function call means, accessing a function. A function can be called by specifying its name, followed by a list of arguments enclosed in the parentheses, and separated by comma

*Example:* f=fact(n);

If a function call does not require any arguments, an empty pair of parentheses must follow the function name.

A function call may appear itself or it may be one of the operand within a expression.

*Example:* printf("%d",fact(n));

Arguments in function call, and formal parameters in function declaration must be same in number, type and order.

### **Difference between actual and formal parameters**

Variables used in function calling are called **actual parameters or arguments**.

Variables used in function definition are called **formal parameters or dummy parameters** or place holders.

There are two parameter passing mechanisms

1. call by value. Or passing by value
2. call by reference or passing by address or call by address.

#### ***Call by value:***

Passing arguments by value means , the contents of the arguments in the calling function are not changed , even if they are changed in the called function.

This is because the content of the variable is copied to the formal parameter of the function

definition, thus preventing the contents of the argument in the calling function.

**Call by reference:**

Call by reference means sending the address of variables as arguments to the function. When addresses are sent, the changes occurred in the called function can also effect in the calling function.

```
/* example to illustrate call by value and call by reference*/
```

```
#include <stdio.h>
void main()
{
void makezero(int,int*); /*function prototype*/
int x=50,y=100;
makezero(x,&y); /* function call by value and referene */
printf("x=%d y=%d", x,y);
}

/* function makezero */
void makezero(int a,int *b)
{
    a=0;
    *b=0;
}
```

**OUTPUT**

```
x=50 y=0
```

**Example programs on functions**

**Factorial of number by using function**

```
#include <stdio.h>
main()
{
long int fact(int);
int n;
long int f;
printf("Enter a number\n");
scanf("%d",&n);
f = fact(n);
printf("The factorial of %d is %ld",n,f);
}/*end of main */

/* function fact*/
```

```

long int fact(int x)
{
    int i;
    long int f1=1;
    for(i=1;i<=x;i++)
        f1 = f1*i;
    return(f1);
}

```

### **Fibonacci series by function**

```

main()
{
    int n;
    void fib(int);
    printf( "\nHow many Fibonacci numbers
required ? ");
    scanf("%d", &n);
    printf("\nThe first %3d fibonaccinumbers  are\n", n);
    fib(n);
}

```

```

void fib(int n)
{
    int i, f1, f2, f3;
    f1 = f2 = 1;
    printf("%5d\n", f1);
    printf("%5d\n", f2);
    for( i = 3; i<= n; i++)
    {
        f3 = f1 + f2;
        printf("%5d\n", f3);
        f1 = f2;
        f2 = f3;
    }
}

```

### **GCD of two numbers by function**

```

void main()
{
    int m,n,g;
    int gcd(int,int);

    printf("Enter two numbers\n");
    scanf("%d%d",&m,&n);
}

```

```

g = gcd(m,n);
printf("GCD of %d and %d is %d", m,n,g);

} /* End of main() */
int gcd(int m,int n)
{
    int r,g1;
    do
    {
        r=m%n;
        if(r!=0)
        {
            m=n;
            n=r;
        }
    }while(r!=0);
    g1=n;
    return(g1);
} /* End of function gcd */

```

### **LCM of two numbers by function**

```

main()
{
    int m,n;
    int lcm(int,int);

    printf("Enter two numbers\n");
    scanf("%d%d",&m,&n);
    printf("LCM of %d and %d is %d", m,n,lcm(m,n));

} /* End of main() */

int lcm(int m,int n)
{
    int i;
    for(i=1;i<= m*n;i++)
    {
        if(i%m==0 &&i%n==0)
            break;
    }
    return i;
} /* End of function lcm*/

```

### **Sum of digits by function**

```

main()
{
    int digitsum(int);
    int num;
    printf("Enter a number\n");
    scanf("%d",&num);
    printf("Sum of digits = %d\n",digitsum(num));
}
/* function digitsum() */
int digitsum(int num)
{
    int digit,sum=0;
    while(num !=0)
    {
        digit = num%10;
        sum = sum+ digit;
        num = num/10;
    }
    return sum;
}

```

### **To check whether the number is palindrome or not by using function**

```

#include <stdio.h>
void main()
{
    int palin(int);
    int num,ans;
    printf("Enter a number\n");
    scanf("%d",&num);
    ans=palin(num);
    if(ans==1)
        printf("%d is palindrome\n",num);
    else
        printf("%d is not palindrome\n",num);
}

/* function palin */
int palin(int num)
{
    int rev=0,temp,digit;
    temp=num;
    while(num!=0)
    {
        digit= num%10;
        rev = rev*10 + digit;
        num = num/10;
    }
    if(rev==temp)

```

```
        return 1;
    else
        return 0;
}
```

## **RECURSIVE FUNCTIONS**

*A function which calls itself until a certain condition is reached is called recursive function*

**/\* to find factorial of a number recursively \*/**

```
#include <stdio.h>
main()
{
    int rfactorial(int);
    int n,f;
    printf("Enter a number\n");
    scanf("%d",&n);
    f=rfactorial(n);
    printf("Factorial of %d is %d",n,f);

}
```

**/\* factorial \*/**

```
int rfactorial(int n)
{
    if(n==0)
        return 1;
    else return(n*rfactorial(n-1));
}
```

**/\* to find nth fib number recursively \*/**

```
#include <stdio.h>
main()
{
    int rfib(int);
    int n,f;
    printf("Enter a number\n");
    scanf("%d",&n);
    f=rfib(n);
    printf("n thfibonacci number is %d",f);
}
```

**/\* rfib \*/**

```
int rfib(int n)
{
    if(n==1 || n==2)
        return 1;
    else return(rfib(n-1)+rfib(n-2));
}
```

**/\* to find GCD of two numbers recursively \*/**

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

```

main()
{
int rgcd(int,int);
int m,n,g;

printf("Enter two numbers\n");
scanf("%d%d",&m,&n);
g=rgcd(m,n);
printf("gcd is %d",g);

}
/*function rgcd */
int rgcd(int m,int n)
{
    if(m%n==0)
        return n;
    else
        return(rgcd(n,m%n));
}

/* to find nCr recursively */
#include <stdio.h>
#include <conio.h>
void main()
{
    int rncr(int,int);
    int n,r,s;
    printf("Enter two numbers\n");
    scanf("%d%d",&n,&r);
    s=rncr(n,r);
    printf("nCr=%d\n",s);
}
int rncr(int n,int r)
{
    if(r==0 || n==r)
        return 1;
    else
        return(rncr(n-1,r)+rncr(n-1,r-1));
}

/* to find sum of digits  recursively */
#include <stdio.h>
#include <conio.h>
void main()
{
    int rsum(int n);
    int n,s;
    printf("Enter a number\n");
    scanf("%d",&n);
}

```

```

s=rsum(n);
printf("sum of digits =%d\n",s);
}

int rsum(int n)
{
    if(n==0)
        return n;
    else return(n%10+rsum(n/10));
}

/* to find m power n recursively */

#include <stdio.h>
void main()
{
    int rpower(int,int);
    int m,n;
    printf("Enter two number\n");
    scanf("%d%d",&m,&n);
    printf("%d power %d is %d\n",m,n,rpower(m,n));

}
int rpower(int m,int n)
{
    if(n==0)
        return 1;
    else
        return(m*rpower(m,n-1));
}

/* to find the sum of two numbers recursively */
#include <stdio.h>
#include <conio.h>
void main()
{
    int rsum(int,int);
    int m,n,s;

    printf("Enter two numbers\n");
    scanf("%d%d",&m,&n);
    s=rsum(m,n);
    printf("sum =%d\n",s);
}

int rsum(int m,int n)
{
    if(n==0)

```

```

        return m;
    else
        return(rsum(m+1,n-1));
}

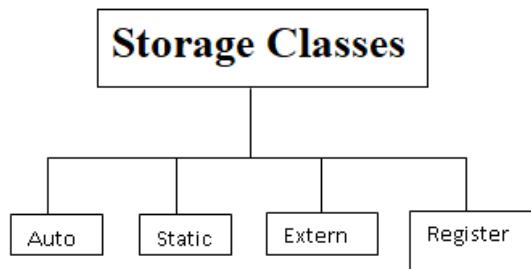
```

## STORAGE CLASSES

Normally the life of a variable is limited to a function as long as the function is alive. How to make it alive in a file or throughout the program or limiting only to a block inside a function or to make common to a desired couple of functions etc., The answer lies in “STORAGE CLASSES” or “VARIABLE TYPES”.

**There are four types of storage classes.**

1. Automatic variables.
2. External variables.
3. Static variables.
4. Register variables.



### AUTOMATIC VARIABLES:

Automatic variables are declared inside a function, in which they are to be utilized. They are created when the function is called and destroyed automatically when the function is exited. By default all variables are automatic variables. These are also called local or internal variables.

### EXTERNAL VARIABLES

Variables that are both alive and active throughout the entire program are called external variables. These variables are available to all functions in that program. Whatever changes that occur in a function, will effect the value of that variable.

### STATIC VARIABLES

When a variable is declared as a static variable it is assigned the value zero.

Static variables are initialized only once. They will not be initialized for second time during the program. When static is applied to a global variable, the global variable becomes inaccessible outside the file.

### **REGISTER VARIABLES**

This is local to a function or a block. If a compiler finds a physical register in the CPU free for the time being, and also big enough to hold the value, then it may stick that variable in that register. Otherwise, the compiler treats that variable as ordinary. It is machine dependent. But compiler will not give error messages even if no register available in reserve. If we know that a particular variable will be used often inside function-say the variable inside a loop-then we can declare that variable with class register. So that the interaction will be very fast.

**//Write a program to demonstrate life cycle of auto variable in nested blocks.**

```
void main()
{
    auto int x=1;
    {
        auto int x=2;
        {
            auto int x=3;
            printf("%d\n", x);
        }
        printf("%d\n",x);
    }
    printf("%d\n",x);
}
```

Output  
3  
2  
1

**// program to illustrate automatic and external variables**

```
#include <stdio.h>
int x=1;
int fun1()
{
    x=x+10;
    return x;
}

int fun2()
{
```

```

int x;
x=20;
return x;
}
int fun3()
{
x=1000;
return x;
}
void main()
{
    int x=100;
    printf("% d\n",fun1());
    printf("%d\n",fun2());
    printf("% d\n",fun3());
    x=0;
    printf("%d\n",x);
printf("% d\n",fun1());
    printf("%d\n",fun2());
    printf("% d\n",fun3());
}

```

OUTPUT

```

11
20
1000
0
1010
20
1000

```

*// program to illustrate static variable*

```

#include <stdio.h>
void main()
{
    int fun();
    int i;
    for(i=1;i<5;i++)
        printf("%d\n",fun());
}

```

```

int fun()
{
    static int x=0;
    x+=1;
    return(x);
}

```

OUTPUT

```

1

```

```

2
3
4
// program to illustrate register variable
#include <stdio.h>
void main()
{
register inti;
int i;
for(i=1;i<1000;i++)
    printf("% d",i);
}

```

<b>automatic variable</b>	<b>static variable</b>
1. It is declared by auto int a = 10; 2. It must be declared inside the function 3. If it is not initialized, garbage value is stored 4. It is initialized every time function is called 5. It is lost when function terminates	1. It is declared by static int a = 10; 2. It can be declared inside or outside the function 3. If it is not initialized, zero is stored 4. It is initialized once and only one but not every time when function is called 5. It is lost when program terminates but not when function terminates

<b>Local variable</b>	<b>Global variable</b>
-----------------------	------------------------

<p>1. It is declaration inside the function</p> <p>2. It is declared by auto int a=10 or int a = 10 (auto is default scope)</p> <p>3. If it is not initialized. garbage value is stored Eg: int a; a is garbage value</p> <p>4. It is created when the function starts execution and lost when the function terminates</p> <p>5. It is visible in only one function.</p> <p>6. It can be accessed in only one function that is the function where it is declared</p> <p>7. Data sharing is not possible that is data of local variable can be accessed by only one function</p> <p>8. Parameters passing is required for local variable that is local variable of one variable</p> <p>9. If value of local variable is modified in one function changes are not visible in another functions</p>	<p>1. It is declaration outside the function</p> <p>2. It is declared by int a=10</p> <p>3. If it is not initialized zero is stored Eg: int a; a = 0</p> <p>4. It is created before program execution starts and lost when program terminates</p> <p>5. It is visible through out the program.</p> <p>6. It can be accessed in more than one function.</p> <p>7. Data sharing is possible that is multiple functions can access the same global variable.</p> <p>8. Parameters passing is not required for global variable since global variable is visible throughout the program</p> <p>9. If value of global variable is modified in one function changes are visible in rest of the program</p>
--	---

Storage Class	Type	Default initial value	Place of declaration	Scope	life
auto	local	Garbage value	Within function or block	Only within the function or block where it is declared	Until function block is no longer active
static	local	zero	"	"	Until program ends
extern	global	zero	Head of all functions within file	All files including other files where declared extern	While any file is active.

**Iterative**

**Recursive**

<p>1. Function calls some other functions</p> <p>2. while loop, do while loop or for loop is necessary in iterative function</p> <p>3. Fast in execution since one function call is enough to get the result</p> <p>4. Function call leads to value Eg: fact(4) = 24</p> <p>5. We require mathematical steps or procedure to write an iterative function.</p> <p>6. Stacks are not used during execution</p>	<p>1. Function calls some itself</p> <p>2. if statement is necessary in recursive function</p> <p>3. Slow in execution since several function calls are involved to get the result, as number of function increases execution will slow down.</p> <p>4. Function call leads to another function call Eg: fact(4) = 4 x fact(3)</p> <p>5. We require a formula to write a recursive function.</p> <p>6. Stacks are used during execution of recursive functions</p>
--	--

### Summary:

- All C programs contain main() function which is mandatory.
- main() function is the function from where every C program is started to execute.
- Name of the function is unique in a C program.
- C Functions can be invoked from anywhere within a C program.
- There can any number of functions be created in a program. There is no limit on this.
- There is no limit in calling C functions in a program.
- One function can be called within another function.
- C functions can be called with or without arguments/parameters. These arguments are nothing but inputs to the functions.
- C functions may or may not return values to calling functions. These values are nothing but output of the functions.
- When a function completes its task, program control is returned to the function from where it is called.
- There can be functions within functions.
- Before calling and defining a function, we have to declare function prototype in order to inform the compiler about the function name, function parameters and return value type.
- C function can return only one value to the calling function.
- When return data type of a function is “void”, then, it won’t return any values
- When return data type of a function is other than void such as “int, float, double”, it

returns value to the calling function.

**Review and revision:**

**MCQs:**

1. In C, parameters are always(L2)(CO4)
  - a) Passed by value
  - b) Passed by reference
  - c) Non-pointer variables are passed by value and pointers are passed by reference
  - d) Passed by value result

**Answer: a**

2. A function prototype is used for (L2) (CO4)
  - a) Declaring the function logic
  - b) Calling the function from the main body
  - c) Telling the compiler, the kind of arguments used in the function
  - d) Telling the user for proper use of syntax while calling the function

**Answer: c**

3. The variables that are declared inside a function are (L1) (CO4)
  - a) global variable
  - b) local variables
  - c) static variables
  - d) must be same as the variables declared in main().

**Answer: b**

**4.**

What is the output of the following C program?

```
#include <stdio.h>
void foo(), f();
int main()
{
    f();
    return 0;
}
void foo()
{
    printf("2 ");
}
void f()
{
    printf("1 ");
    foo();
}
```

(L3) (CO4)

- a) Compiler error as foo() is not declared in main
- b) 1 2
- c) 2 1
- d) Compile time error due to declaration of functions inside main

**Answer: b**

5. Can we use a function as a parameter of another function? [Eg: void func1(int func2())]? (L1) (CO4)
- a) Yes, and we can use the function value conveniently
  - b) Yes, but we call the function again to get the value, not as convenient as in using variable
  - c) No, C does not support it
  - d) This case is compiler dependent

**Answer: c**

6. What is the return-type of the function sqrt().(L2) (CO4)
- a) int
  - b) double
  - c) float
  - d) depends on compiler

**Answer: b**

7. What is the default return type if it is not specified in function definition??(L3) (CO4)
- a) void
  - b) integer
  - c) double
  - d) float

**Answer: b**

8.

Identify What is the error in the following program

```
#include<stdio.h>
int f(int a)
{
    a > 20? return(10): return(20);
}
int main()
{
    int b;
    b = f(20);
    printf("%d\n", b);
    return 0;
}
```

(L3) (CO4)

- a) Error: Return statement cannot be used with conditional operators
- b) Error: Prototype declaration
- c) Error: Two return statements cannot be used in any function
- d) No error

**Answer: a**

9. Identify which statement is correct about Passing by value parameters.(L3)(CO4)

- a) It cannot change the actual parameter value
- b) It can change the actual parameter value
- c) Parameters is always in read-write mode
- d) None of them

**Answer: a**

10. Which keyword is used to give back the value?(L5) (CO4)

- a) static
- b) void
- c) return
- d) const

**Answer: b**

11. Evaluate the output of the following program. (L5) (CO4)

```
#include <stdio.h>
void foo()
{
    return 1;
}
void main()
{
    int x = 0;
    x = foo();
    printf("%d", x);
}
```

- a)Compile time error
- b)0
- c)1
- d) run time error

**Answer: a**

**12.** Number of values a functionin C can return? (L2) (CO4)

- a)2
- b)0
- c)1
- d) 3

**Answer: c**

**13.** Types of functions in C language(L2) (CO4)

- a)User defined
- b)Library
- c)both (a) and (b)
- d) None

**Answer: c**

**14.** C program must contain at least(L2) (CO4)

- a) n functions
- b)1 function
- c) 2 functions
- d) None

**Answer: b**

**15.** which of the following is not a storage class specifier?(L2) (CO4)

- a) auto
- b) register
- c) extern
- d) volatile

**Answer: d**

**16.** what is the initial value of register storage class specifier? (L2) (CO4)

- a)0
- b)garbage
- c)1
- d)infinite

**Answer: b**

**Review and revision exercise (2) :**

Case Study:

1. To display the given decimal number in words using user defined function
2. To display the given decimal number in roman numbers using user defined function
3. To find the day name when date is entered using user defined function

**Expected Questions:**

**Short answer type:**

1. What is a function? What is the need for functions?(L1) (CO4)
2. Classify different types of functions. (L2) (CO4)
3. Explain the general form of defining a function. (L2) (CO4)
4. Define i) function prototype ii) function header iii) recursive function (L1) (CO4)
5. Differentiate between (L2) (CO4)
  - i) actual and formal parameters
    - ii) calling function and called function
    - iii) user define function and library function
    - iv) call by value and call by reference
    - v) function prototype and function header
    - vi) local and global variables
  6. Write about storage classes. (L1) (CO4)
  7. What are different types of variables? Explain with examples. (L1) (CO4)

**Long answer type:**

8. Develop a C program to find factorial of a number? (L3) (CO4)
9. Develop a C program to find nth Fibonacci number? (L3) (CO4)
10. Develop a C program to find GCD of two numbers? (L3) (CO4)
11. Develop a C program to find sum of digits of a number? (L3) (CO4)
12. Develop a C program to find prime or not? (L3) (CO4)
13. Develop a C program to find palindrome or not? (L3) (CO4)
14. Develop a C program to find Fibonacci series? (L3) (CO4)
15. Develop a C program to find power of a number by using non recursive and recursive functions? (L3)(CO4)



# CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. [www.cblt.ac.in](http://www.cblt.ac.in)

Established In  
1979



Accredited by  
NAAC



With Rank In  
NAAC



COMMITTED TO  
RESEARCH,  
INNOVATION AND  
EDUCATION

41  
years

**Course Code: 20CS C01**

**Course Title: PROGRAMMING FOR PROBLEM SOLVING**

**UNIT-III**

**Objective:**

The primary goals of the module: To understand:

- Arrays and different dimensional arrays and their needs.
- One dimensional and two dimensional arrays declaration, representation in memory, accessing the data and performing operations on them.
- call by reference mechanism involved in array, passing to a function

**Outcome:**

On Successful completion of the module: Arrays, students will be able to:

- Learn about one-dimensional and two-dimensional arrays and the way they are declared, initialized, manipulated, inputted, and displayed.
- Differentiate different dimensions of the array and able to choose appropriate array to store data and perform operations on them in an application.

**Scope:**

The Module covers the following topics: Introduction to Arrays, Declaration using Arrays in C, Accessing and storage of array elements, one dimensional arrays, Searching Algorithms (linear and binary search), Sorting Algorithms (Selection and Bubble), Two dimensional arrays, matrix operations.

While these topics are taught using a C language, the intent of the course is to teach a programming methodology and also a laboratory component that involves development and testing of iterative and procedural programs using one-dimensional and two-dimensional arrays.

**Text and Reference:**

**(a) Text Book:**

1. M.T. Somashekhar "Problem Solving with C", 2 nd Edition, Prentice Hall India Learning Private Limited 2018
2. A K Sharma "Computer Fundamentals and Programming", 2nd Edition, University Press, 2018
3. Pradeep Dey and Manas Ghosh, "Programming in C", Oxford Press, 2nd Edition, 2017

**(b) Reference Books:**

1. Byron Gottfried, Schaum's "Outline of Programming with C", McGraw- Hill.
2. Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Prentice Hall of India.
3. E. Balaguruswamy, Programming in ANSI C, Tata McGraw-Hill.
4. ReemaTharaja "Introduction to C Programming", Second Edition, OXFORD Press, 2015.
5. <https://www.tutorialspoint.com/cprogramming/index.htm>.
6. <https://onlinecourses.nptel.ac.in/noc18-cs10/preview>.

Module/Lecture	Theme	Learning Objectives
L.19	Introduction to Arrays, Declaration, Using Arrays in C	To learn about the need of arrays and its representation. Its declaration syntax
L.20	Accessing and storage of array elements, one dimensional arrays	To learn the memory allocation for arrays and accessing elements. Perform basic operations on array elements
L.21	Searching Algorithms (linear and binary search)	Using search operation learner will learn accessing element and apply operators on array elements
L.22	Sorting Algorithms (Selection and Bubble)	Learn how nested loops can be used to access elements in the arrays to perform sorting and analysis each passes.
L.23	Two dimensional arrays, matrix operations	Learn how multiple dimensional can be declared, initialized, accessed and performing operations on them

Lecture Title:	L19 to L23
----------------	------------

Date	
------	--

Key topics : Integer one dimensional and multi-dimensional arrays and its storing, retrieving and memory locations

### Introduction to Arrays

Till now you might have declared variables to hold data and to perform operations on it. Just like given below-

```
int a=10;  
char ch='a';
```

These variables can hold only one data item at a time. Let us take some example where collection of data items need to be stored....

- 60 students marks secured in a subject
- Salaries of 100 employees

we are supposed to declare multiple variables to hold the above data for each example.

language provides a derived data types called arrays.

#### definition

An array is collection of homogeneous elements stored in continuous memory locations and shares a common name.

ere are several forms of an array used in C: one-dimensional or single-dimensional and multidimensional array.

#### declaration of one-dimensional or single-dimensional

Like any other variables, arrays must be declared before they are used. The general form of array declaration is

**Syntax:** datatype array\_name[sizeofarray];

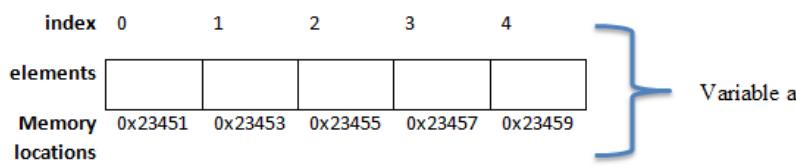
- The data type specifies the type of data/element that will be stored in the array, such as int, float, or char.
- The size indicates the maximum number of data/elements that can be stored in the array.
- The size of array should be a int constant or constant expression.

Let us take the same example to declare suitable one dimensional array variable to hold the data.

- 60 students marks secured in a subject  
*unsigned short int PPS[60];*
- Salaries of 100 employees  
*float salaries\_of\_employees[100];*

#### ernal representation /Memory Allocation

a[5];



- Elements are stored in consecutive memory allocations. Each block is of size two because variable 'a' is of data type 'int'. If array variable is of 'float' data type then each block occupies four bytes.
- So total memory occupied by an array whose size is given as 'n' is-  $\text{sizeof}(\text{datatype of the variable}) * n$
- First element in the array is stored at  $a[0]$ , second at  $a[1]$  and so on fifth element is stored at  $a[4]$ . If size of an array is 'n' then elements in the array stored from 0 to  $n-1$  index/subscript. Storing/retrieving data into/from the array is done by using index/subscript.

### Processing one-dimensional array elements

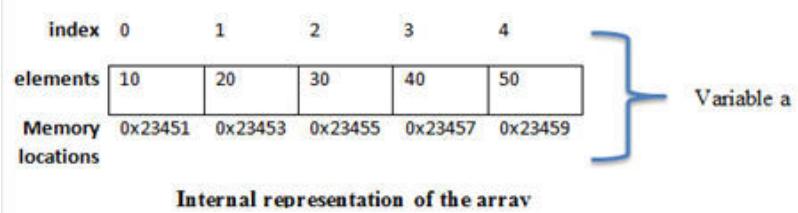
Array access involves store/initialize and retrieve the data items.

- Array is initialized with elements can be done at static time (while writing the code) or dynamic time( at the time of executing the program).

#### Static Initialization

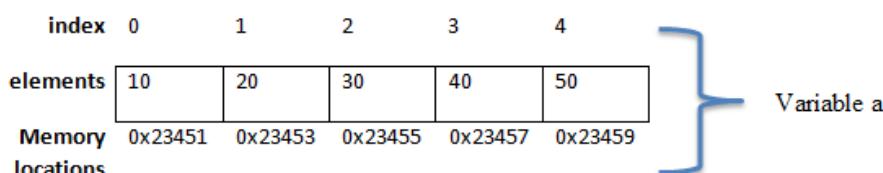
##### **Example 1:**

```
int a[5]; //array declaration
//array initialization
a[0]=10;
a[1]=20;
a[2]=30;
a[3]=40;
a[4]=50;
```



##### **Example 2:**

```
int a[5]={10,20,30,40,50}; // array declaration and initialization
```



#### Dynamic Initialization

##### **Example 3:**

```
int a[5],i;
printf("Enter 5 integer elements:");
for(i=0;i<=4;i++)
```

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

At the time of execution user can give 5 data items which will be stored in the array.

```
Enter 5 integer elements:12 33 44 55 66
```

index	0	1	2	3	4	
elements	12	33	44	55	66	
Memory locations	0x23451	0x23453	0x23455	0x23457	0x23459	



Variable a

#### Internal representation of the array

##### Retrieving elements from an array using subscript

```
int a[5]={10,20,30,40,50};  
printf("%d\t%d\t%d\t%d\t%d",a[0], a[1], a[2], a[3], a[4]);
```

##### Retrieving elements from an array using loop/iterative variable as subscript

```
int a[5]={10,20,30,40,50},i;  
for(i=0;i<=4;i++)  
printf("%d",a[i]);
```

In both the cases elements stored in the indexes from '0' to '4' will be displayed.

As loop variable value changes the subscript of an array changes and corresponding element will be displayed.

#### Let us write a program to store and retrieve element into/from an array.

```
# include<stdio.h>  
int main( )  
{  
inti , a[5];  
printf("Enter 5 integer elements: "); /* reading values into Array */  
    for (i=0; i<5; i++)  
        scanf(" %d ", &a[i]);  
    /* printing of a[i] values */  
    printf("The array elements are:");  
    for(i=0; i< 5; i++)  
        printf(" %d ", a[i]);  
}
```

**Output:**

```
Enter 5 integer elements:33 12 77 65 32
The array elements are:33 12 77 65 32
```

### Variable length arrays

Previous version of C(C89) compilers, an array's size must be a constant, so that compiler can calculate the size/memory allocation at the compilation time itself. But from C99 onwards, C compilers allows the programmers to declare an array variable with variable length ie size of an array can be given at execution time/ run time.

```
/* Program to read and display using variable length arrays*/
```

```
int main()
{
intn,i;
printf("\n Enter the size of the array");
scanf("%d",&n);
int a[n]; // variable length array (VLA). The 'n' variable declaration should be before VLA
printf("\n Enter %d numbers:",n);
for(i=0;i<=n-1;i++)
scanf("%d",&a[i]);
printf("\n\n Given %d numbers are:",n);
for(i=0;i<n;i++)
printf("%d ",a[i]);
return 0;
}
```

#### Output:

```
Enter the size of the array10
Enter 10 numbers:11 33 44 -77 -66 99 11 99 2 1
Given 10 numbers are:11 33 44 -77 -66 99 11 99 2 1
```

### Operations on array elements

Some operations are listed below

- $c=a[0]+a[1]$  // adding elements at index '0' and '1' and store it in 'c' variable
- $a[i]=a[i]+20$  // adding a constant to the array element at index 'i' and store in same location
- $a[i]>x$  // compare value at 'x' with value at 'a[i]'
- $a[i]=a[i]+1$  // add '1' to value at 'a[i]'
- $a[i]++$  // add '1' to value at 'a[i]'
- $a[i]+=1$  // add '1' to value at 'a[i]'
- $a[k]=a[i];$  // 'i'th indexed element stored at index 'k'

```
/* Write a program to print the array elements in reverse order */
```

```
# include<stdio.h>
int main( )
```

```

{
inti , a[5];
printf("Enter 5 integer elements: "); /* reading values into Array */
    for (i=0; i<5; i++)
        scanf(" %d ", &a[i]);
    /* printing of a[i] values */
    printf("The array elements in reverse order are:");
        for(i=4; i>=0 ; i--)
printf("%d ", a[i]);

}

```

**Output:**

```

Enter 5 integer elements:22 44 22 11 66
The array elements in reverse order are:66 11 22 44 22

```

### Searching operation on array elements

Find whether a given element/key exists or not in the array of elements is called search. This operation can be done by using two methods- simplest method is a sequential search or linear search and binary search.

The basic idea of **linear search** is start comparing the key with each element in the array from index '0' to index 'n'-1 if 'n' is the size of the array.

if key is found display the position/index at which element exist and exit the search otherwise search will be continued till the end of the array and display the key is not found.

```

/* Program to implement linear search */
#include <stdio.h>
#include <stdlib.h>
int main()
{
int a[10],n,i,key, FOUND=0;
printf("\n How many numbers");
scanf("%d",&n);
if(n>10)
{
    printf("\n Too many Numbers");
    exit(0);
}
printf("\n Enter the array elements \n");
for(i=0 ; i<n; i++)
    scanf("%d", &a[i]);
printf("\n Enter the key to be searched \n");

```

```

scanf("%d",&key);
for(i=0 ; i<n; i++)
    if(a[i] == key)
    {
        printf("\n Found at %d",i);
        FOUND=1; break;
    }
if(FOUND == 0)
    printf("\n NOT FOUND...");
return 0;
}

```

**Output when more than the size of the array**

```

How many numbers12
Too many Numbers

```

Output when search is successful	Logic for successful search
<pre> How many numbers5 Enter the array elements 33 44 -55 77 31 Enter the key to be searched 31 Found at 4 </pre>	<pre> Iteration 1 compare 33 and 31 Iteration 2 compare 44 and 31 Iteration 3 compare -55 and 31 Iteration 4 compare 77 and 31 Iteration 5 compare 31 and 31 Found at 4 </pre>
<b>Output when search is unsuccessful</b> <pre> How many numbers5 Enter the array elements 33 66 11 -33 12 Enter the key to be searched 10 NOT FOUND... </pre>	<b>Logic for unsuccessful search</b> <pre> Iteration 1 compare 33 and 10 Iteration 2 compare 66 and 10 Iteration 3 compare 11 and 10 Iteration 4 compare -33 and 10 Iteration 5 compare 12 and 10 NOT FOUND... </pre>

In **binary search method** , halves the size of the list and searches only in half of the array by finding the mid element index using first(low) and last(high) index of the array.

The procedure is as follows-

- Find mid element by using low and high
- If key is less than mid element then search will be done in left of the array by updating high
- Else If key is greater than mid element then search will be done in right of the array by updating low
- Otherwise mid element itself is the search element.

It will be continued until low is less than high. Unsuccessful search make low greater than high. To perform binary search the elements should be increasing order in the array.

```

/* Program to implement linear search */

#include<stdio.h>
int main()
{
int n;
printf("\n Enter the size of the array:");
scanf("%d",&n);

int a[n],i,key,FOUND=0;

printf("\n Enter elements in increasing order");
for(i=0;i<n;i++)
scanf("%d",&a[i]);

printf("\n Enter element for searching:");
scanf("%d",&key);

int low=0,high=n-1,mid;
while(low<=high)
{
    mid=(low+high)/2;
    if(key>a[mid])
        low=mid+1;
    else if(key<a[mid])
        high=mid-1;
    else
    {
        printf("\n Found at %d",i);
        FOUND=1; break;
    }
}
if(FOUND == 0)
    printf("\n NOT FOUND...");

return 0;
}

```

#### Output

```

Enter the size of the array:5
Enter elements in increasing order1 2 3 4 5
Enter element for searching:5
Found at 5

```

#### Binary search logic Interpretation for successful search

```

Indexes at low =0 high=4 mid=2
3 compared with 5 and updated low=3
Indexes at low =3 high=4 mid=3
4 compared with 5 and updated low=4
Indexes at low =4 high=4 mid=4
Found at 5

```

### Bubble Sorting

Sorting operation will keep the element in an order.

Basic principle of bubble sort is consecutive/adjacent elements are compared and exchanges their values if they are not in the order. This process bubbles the smallest element in the array to '0' index and largest to 'n-1' index where 'n' is the size of the array.

The following depicts the different stages of the bubble sort.

Array elements are 44 22 -1 3 10

pass1					Comparison on
44	22	-1	3	10	44 , 22 (swap)
22	44	-1	3	10	44, -1 (swap)
22	-1	44	3	10	44 , 3 (swap)
22	-1	3	44	10	44, 10 (swap)
22	-1	3	10	44	Pass1 completed and largest element (44) placed in the correct position. It is not going to be considered in the next iteration.
Pass2					Comparison on
-1	22	3	10	44	22, -1 (swap)
-1	3	22	10	44	22, 3 (swap)
-1	3	10	22	44	22, 10 (swap)
-1	3	10	22	44	Pass 2 completed . 22 is placed in the correct position. It is not going to be considered in the next iteration.
Pass3					Comparison on
-1	3	10	22	44	-1, 3 (no swap)
-1	3	10	22	44	3,10 (no swap)
-1	3	10	22	44	Pass 3 completed . 10 is placed in the correct position .It is not going to be considered in the next iteration.
Pass4					Comparison on
-1	3	10	22	44	-1, 3 (no swap)
-1	3	10	22	44	Pass 4 completed . 3 is placed in the correct position .It is not going to be considered in the next iteration.

Always single element in the array will be in sorted order itself. So algorithm stops its passes.

```
/* Program to implement bubble sort*/
#include<stdio.h>
int main()
{
int n;
printf("\n Enter the size of the array:");
scanf("%d",&n);
int a[n],i,j,temp;
printf("\n Enter %d numbers for sorting",n);
for(i=0;i<n;i++)
scanf("%d",&a[i]);
printf("\n Before Sorting ....\n");

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

```

printf("%5d",x[i]);

//sorting logic
for(i=0;i<n-1;i++)
{
    for(j=0;j<n-1-i;j++)
    {
        if(a[j]>a[j+1])
        {
            temp=a[j];
            a[j]=a[j+1]; /* swap */
            a[j+1]=temp;
        }
    }
}

printf("\n After Sorting ....\n");
for(i=0;i<n;i++)
printf("%5d",x[i]);
return 0;
}

```

### Selection Sorting

- Selection sort finds the minimum element from the array and place it in the beginning.

Array elements are 44 22 -1 3 10

44	22	-1	3	10	<b>Pass 1:</b> Find minimum element from index 0 to 4 and place at index 0 ie min=-1 Swap( 44,-1)
-1	22	44	3	10	<b>Pass 2:</b> Find next minimum element from index 1 to 4 and place at index 1 ie min=3 Swap( 22,3)
-1	3	44	22	10	<b>Pass 3:</b> Find next minimum element from index 2 to 4 and place at index 2 ie min=10 Swap( 44,10)
-1	3	10	22	44	<b>Pass 4:</b> Find next minimum element from index 3 to 4 and place at index 3 ie min=22 Swap( 22,22)
-1	3	10	22	44	Single element will be in sorted order

/\* Program to implement Selection Sort \*/

```

#include <stdio.h>
main()
{
int a[20],n,i, j, po,se,temp;
printf("How many elements\n");
scanf("%d",&n);

/* loop to read values into array */
printf("Enter %d numbers below:\n", n);
for ( i = 0; i < n; i++ )
scanf("%d", &a[i]);

/* loops of selection sort */
for(i=0;i<n-1;i++)
{
    se=po=i;
    for(j=i+1;j<n;j++)
    {
        if(a[j] < a[se])
            se=j;
    }
    temp=a[po];
    a[po]=a[se];
    a[se]=temp;
}

/* loop to display sorted array */
printf("The sorted Array is: \n");
for ( i = 0; i < n; i++ )
printf("%d\n", a[i]);

}

```

### **TWO DIMENSIONAL ARRAYS**

- Arrays with more than one dimension are called multidimensional arrays.
- Array of one dimensional array is called two dimensional array.

An array of two dimensions can be declared as follows:

#### **Syntax:**

**data\_typearray\_name[size1][size2];**

- size1, size2 are constant , array subscript in both dimensions start from 0 to size1-1 and 0 to size2-1. Size1 tells the information about number of rows and size2 tells about number of columns .
- In other words, size1 gives how many one-dimensional arrays exists and size2 gives how many elements in a one dimensional array.
- Number of elements stored in the array will be size1 \*size 2 .

#### **Example:**

```
int a[10][10];
float b[20][20];
```

- Two dimensions are useful to store data in a table form. Ie take the previous example 60 student marks in a subject.
- In order to store 60 students' marks obtained in 6 subjects then six one dimensional arrays need to be declared for each subjects.
- It will be difficult to remember the variable name of each subject if multiple subjects exist.

#### Static initialization

*initialization of an array:* this can be done in two ways

1. int a[2][2] = { {3,6} , {2,9} }; // each 1D array enclosed in {} and separated with ,
2. int a[2][2] = { 3,6,2,9}; // based on 2<sup>nd</sup> dimension it will identify the number of // elements in each row.

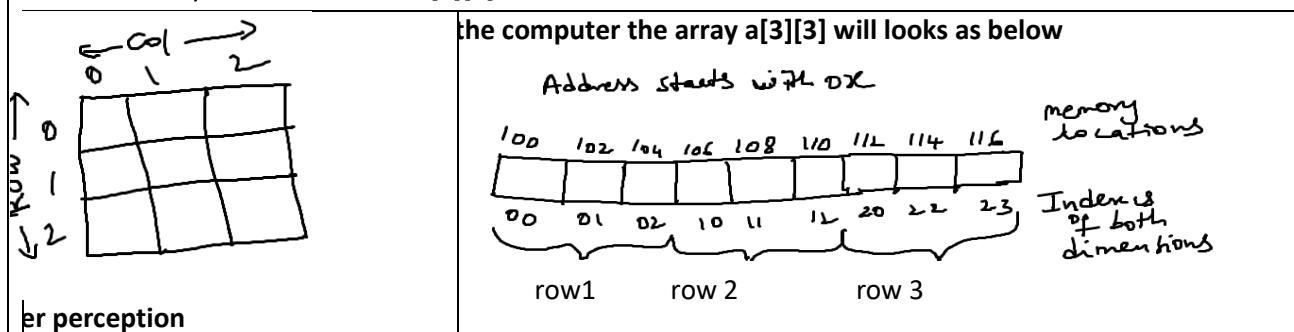
#### dynamic initialization

- To represent 2 dimensions we need two variable one for each dimension.

```
int a[3][3],i,j;
for(i=0;i<3;i++)
{
    for(j=0;j<3;j++)
        scanf("%d",&a[i][j]);
}
```

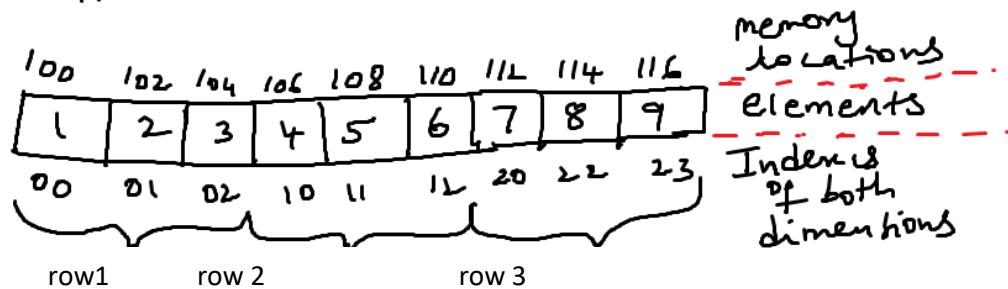
#### **Internal representation /Memory Allocation**

- If array is declared as int a[3][3]



```
a[3][3]={1,2,3,4,5,6,7,8,9};
en the array will be ...
```

## Address starts w/ L DX



- To access the 2<sup>nd</sup> element at 3<sup>rd</sup> row we need to use the indexes as a[1][2]. So element in i<sup>th</sup> row j<sup>th</sup> column will be accessed as a[i-1][j-1].
- All the elements are stored in continuous locations. Two dimensional array is also called as array of arrays.
- First dimension in the 2D array tells about how many one dimensional arrays are required, second dimension tells about how many elements in each one dimensional array .

### Program to add two M x N matrices.

```
#include <stdio.h>
main()
{
int a[10][10],b[10][10],c[10][10],i,j,m,n;

printf("Enter the number of rows and
       columns for A and B\n");
scanf("%d%d",&m,&n);
/* loop to read values of matrix A*/
printf("Enter the elements for matrix A\n");
for(i=0;i<m;i++)
    for(j=0;j<n;j++)
        scanf("%d",&a[i][j]);

/* loop to read values matrix B*/
printf("Enter the elements for matrix B\n");
for(i=0;i<m;i++)
    for(j=0;j<n;j++)
        scanf("%d",&b[i][j]);

/* loop to add two matrices*/
for(i=0;i<m;i++)
    for(j=0;j<n;j++)
        c[i][j] = a[i][j] + b[i][j];

/* loop to print resultant matrix */
printf("The sum of A and B is\n");
```

```

for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
printf("%3d",c[i][j]);
printf("\n");
}
}

```

Enter the number of rows and columns for A and B

2

2

Enter the elements for matrix A

1 2 3 4

Enter the elements for matrix B

4 3 2 1

The sum of A and B is

5 5

5 5

#### **Program to multiply two M x N and P x Q matrices**

```

#include <stdio.h>
main()
{
int a[10][10],b[10][10],c[10][10],i,j,m,n,p,q,k;
printf("Enter the number of rows and columns for A \n");
scanf("%d%d",&m,&n);
printf("Enter the number of rows and columns for B \n");
scanf("%d%d",&p,&q);

/* loop to read values of matrix A*/
printf("Enter the elements for matrix A\n");
for(i=0;i<m;i++)
{
    for(j=0;j<n;j++)
scanf("%d",&a[i][j]);

/* loop to read values of matrix B*/
printf("Enter the elements for matrix B\n");
for(i=0;i<p;i++)
{
    for(j=0;j<q;j++)
scanf("%d",&b[i][j]);
/* loop to multiply two matrices*/
for(i=0;i<m;i++)
{
    for(j=0;j<q;j++)
    {
        c[i][j] = 0;
        for(k=0;k<n;k++)
c[i][j] = c[i][j] + a[i][k]*b[k][j];
    }
}
}

```

```

        }
    }

/* loop to display resultant matrix */
printf("The multiplication of A and B is\n");
for(i=0;i<m;i++)
{
    for(j=0;j<q;j++)
printf("%3d",c[i][j]);
printf("\n");
}
}

```

Enter the number of rows and columns for A

2 3

Enter the number of rows and columns for B

3 2

Enter the elements for matrix A

2 2 2

2 2 2

Enter the elements for matrix B

3 3

3 3

3 3

The multiplication of A and B is

18 18

18 18

#### **Program to find Transpose of a matrix**

```

#include <stdio.h>
main()
{
int a[5][5],b[5][5],i,j,r,c;
printf("Enter the size of matrix\n");
scanf("%d%d",&r,&c);

/* loop to read values of matrix A*/

printf("Enter the matrix values\n");
for(i=0;i<r;i++)
    for(j=0;j<c;j++)
scanf("%d",&a[i][j]);

/* loop to find transpose of matrix A*/
for(i=0;i<r;i++)
    for(j=0;j<c;j++)
        b[j][i] = a[i][j];

```

```

/* loop to display the resultant matrix*/
printf("Transpose of a :\n");
for(i=0;i<c;i++)
{
    for(j=0;j<r;j++)
printf("%3d",b[i][j]);
printf("\n");
}
}

```

Enter the size of matrix

2 3

Enter the matrix values

1 2 3

4 5 6

Transpose of a :

1 4

2 5

3 6

## 1 D ARRAYS AND FUNCTIONS

1. We can pass entire array to a function.
2. In the calling function we need to give only name of the array, subscript or square brackets are not necessary.
3. When we pass array name to a function, base address of that array is passed, since array name refers base address of that array. Therefore call by reference mechanism takes place.
4. It is optional to give size of the array in the function definition.

### /\* linear search by function \*/

```

#include <stdio.h>
main()
{
    void lsearch(int [],int,int);
int a[10],n,i,key;
printf("enter how many elements\n");
scanf("%d",&n);
printf("Enter the values\n");
for(i=0;i<n;i++)
    scanf("%d",&a[i]);
printf("enter the key\n");
scanf("%d",&key);
lsearch(a,n,key);
}
/*Function lsearch */

void lsearch(int a[10],int n,int key)
{
int i,found=0;

```

```

for(i=0;i<n;i++)
{
    if(a[i]==key)
    {
        printf("search is successful\n");
        printf("location = %d\n",i+1);
        found =1;
    }
}
if(found==0)
printf(" element not found\n");

```

**OUTPUT:**

Enter how many numbers

8 ↴

Enter the number

34 5 89 54 7 15 77 23 ↴

Enter the key

77 ↴

Search is successful

Location =7

**/\* binary search by function \*/**

```

#include <stdio.h>
main()
{
void input(int [ ],int);
void bsearch(int [],int,int);
int a[25],n,key;
printf("How many numbers ? ");
scanf("%d",&n);
    printf("\nEnter %d numbers :\n", n);
    input(a,n);
    printf("Enter the key\n");
    scanf("%d",&key);
    bsearch(a,n,key);

} /* end of main */
/*Function input */
void input( int a[],int n)
{
    inti;
    for ( i = 0; i < n; i++ )
scanf("%d", &a[i]);
} /* end of function input */

```

```

/* Function Bsearch */
void bsearch (int a[],intn,int key)
{
intlow,high,mid,flag=0;
low=0;
high=n-1;
while(low<=high)
{
    mid=(low+high)/2;
    if(key < a[mid])
        high=mid-1;
    else if(key > a[mid])
        low = mid+1;
    else if(key == a[mid])
    {
        printf("search is successful\n");
        printf("location = %d\n",mid+1);
        flag=1;
        break;
    }
}
if(flag==0)
printf("search is unsuccessful\n");
} /* end of function bubblesort */

```

**OUTPUT:**

Enter how many numbers

10 ↴

Enter the number

3 5 8 54 7 15 76 23 79 11 ↴

Enter the key

79 ↴

Search is successful

Location =9

**/\*bubble sort by functions \*/**

```

#include<stdio.h>
main()
{
    void input(int [],int);
    void output(int [],int);
    void bubblesort(int [],int);

    int a[25],n;
    printf("How many numbers to sort ? ");
    scanf("%d",&n);
}

```

```

printf("\nEnter %d numbers below:\n", n);
input(a,n);
bubblesort(a,n);
printf("\nThe sorted Array is: \n");
output(a,n);
}/* end of main */

/*Function input */
void    input( int a[],int n)
{
inti;
for ( i = 0; i< n; i++)
scanf("%d", &a[i]);
} /* end of function input */

/* Function Bubblesort */
void    bubblesort (int a[],int n)
{
int pass, j, temp;
for ( pass = 1; pass < n; pass++)
{
for ( j =0 ; j < n-1; j++)
{
if( a[j] > a[j+1])
{
temp = a[j];
a[j] = a[j+1];
a[j+1] = temp;
}
}
}
}/* end of function bubblesort */

/* Function output */
void    output (int a[],int n)
{
inti;
for ( i = 0; i< n; i++)
printf("%5d\n", a[i]);
}/* end of function output */

```

### Selection sort by function

```

#include <stdio.h>
main()
{
    void ssprt(int [],int);

```

```

void display(int [],int);
void readarray(int [],int);
int a[20],n;
printf("How many elements\n");
scanf("%d",&n);
printf("Enter the elements\n");
readarray(a,n);
ssort(a,n);
display(a,n);
}

void readarray(int a[],int n)
{
int i;
for(i=0;i<n;i++)
scanf("%d",&a[i]);
}

void ssort(int a[], int n)
{
int i,j,po,se,temp;
for(i=0;i<n-1;i++)
{
    se=po=i;
    for(j=i+1;j<n;j++)
    {
        if(a[j] < a[se])
            se=j;
    }
    temp=a[po];
    a[po]=a[se];
    a[se]=temp;
}
}

void display(int a[],int n)
{
int i;
printf("The sorted array is\n");
for(i=0;i<n;i++)
printf("%d\n",a[i]);
}

```

How many numbers to sort ?

7

Enter 7 numbers below:

55 4 -3 0 22 1 5

The sorted Array is:

-3

```
0  
1  
4  
5  
22  
55
```

## **2 D ARRAYS AND FUNCTIONS**

1. We can pass entire two dimensional array to function like one dimensional array.
2. When we pass two dimensional array to a function, in the function definition formal argument, the size of the column is must and size of the row is optional.

```
/*Program to add two matrices by using functions*/
```

```
#include <stdio.h>  
main()  
{  
    void readmat(int [ ][10],int,int);  
    void matadd(int [ ][10],int [ ][10],int [ ][10],int,int);  
    void display(int [ ][10],int,int);  
  
    int A[10][10], B[10][10], C[10][10],m,n;  
  
    printf("\nEnter size of the matrix A and B: ");  
    scanf("%d%d", &m,&n);  
  
    printf("\nEnter elements of matrix A \n");  
    readmat (A, m,n);  
  
    printf("\nEnter elements of matrix B \n");  
    readmat (B,m,n);  
  
    matadd(A, B, C,m,n);  
  
    printf("Matrix A is :\n");  
    display(A, m,n);  
  
    printf("Matrix B is :\n");  
    display(B, m,n);  
  
    printf("\nMATRIX C (= A + B) is:\n");  
    display(C, m,n);  
}  
/* end of main */
```

```

/* Function to read the elements of a matrix row-wise */
void readmat(int X[10][10],int rows,int cols)
{
int i, j;
    for(i = 0; i < rows; i++)
        for(j = 0; j < cols; j++)
            scanf("%d", &X[i][j]);
} /* end of function read_matrix */

/*Function to add two mxn matrices */
void matadd (int A[10][10], int B[10][10], int C[10][10], int m, int n)
{
int i, j;
    for(i = 0; i < m; i++)
    {
        for(j = 0; j < n; j++)
        {
            C[i][j] = A[i][j] + B[i][j];
        }
    }
} /* end of function matmul */

/*Function to print the elements of a matrix */
void display(int X[10][10],int rows ,int cols)
{
int i, j;
    for(i = 0; i < rows; i++)
    {
        for(j = 0; j < cols; j++)
            printf("%5d", X[i][j]);
        printf("\n");
    }
} /* end of function display */

/*Program to multiply two matrices by using functions*/
#include <stdio.h>
main()
{
    void readmat(int [] [10],int,int);
    void matmul(int [] [10],int [] [10],int [] [10],int,int,int);
    void display(int [] [10],int,int);

    int A[10][10], B[10][10], C[10][10],m,n,p,q;

    printf("\nEnter size of the matrix A: ");
    scanf("%d%d", &m,&n);
    printf("\nEnter size of the matrix B: ");
}

```

```

scanf("%d%d", &p,&q);

if(n!=p)
    printf("multiplication not possible\n");
else
{
printf("\nEnter elements of matrix A \n");
readmat (A, m,n);

printf("\nEnter elements of matrix B \n");
readmat (B,p,q);

matmul(A, B, C,m,q,n);

printf("Matrix A is :\n");
display(A, m,n);

printf("Matrix B is :\n");
display(B, p,q);

printf("\nMATRIX C (= A X B) is:\n");
display(C, m,q);
}
} /* end of main */

/* Function to read the elements of a matrix row-wise */
void readmat(int X[10][10],introws,int cols)
{
inti, j;
for(i = 0; i< rows; i++)
    for( j = 0; j < cols; j++)
scanf("%d", &X[i][j]);
} /* end of function read_matrix */

/*Function to multiply m x n and p x q matrices */
void matmul (int A[10][10], int B[10][10], int C[10][10],
int m, int q, int n)
{
inti, j,k;
for(i = 0; i< m; i++)
{
    for( j = 0; j < q; j++)
    {
        C[i][j] =0;
        for(k=0; k<n; k++)
        {
            C[i][j] = C[i][j] +A[i][k] * B[k][j];
        }
    }
}
}

```

```

        }
    }
}
} /* end of function matmul */

/*Function to print the elements of a matrix */
void display(int X[10][10],int rows ,int cols)
{
int i,j;
for(i = 0; i< rows; i++)
{
    for( j = 0; j < cols; j++)
        printf("%5d", X[i][j]);
    printf("\n");
}
} /* end of function display */

```

#### **Summary exercise:**

1. Arrays hold homogeneous data and share a common name.
2. An array that contains single index is one dimensional array. An integer index is used to access the elements from the range '0' to 'n-1' where 'n' is the array size.
3. Multi-dimensional arrays contain multiple indexes.
4. Array name itself will give the base address of an array. So when a one dimensional array is passed to function, call by reference mechanism takes place.
5. When two dimensional array is passed to function, in the function definition formal argument, the size of the column is must and size of the row is optional.

#### **Review and revision:**

##### **Review and revision exercise (1) :**

##### **ture topic quiz / MCQs: (All these quiz's belongs to BL3 & BL4)**

1. Test for the output of the following program, if the array begins at address 1200?

```

main()
{
    intarr[] = {2,3,4,1,6};
    printf("%d %d",arr, sizeof(arr));
}

```

2. Identify that the array name gives the base address in all the contexts?

3. Test for the output of the following program, if the array begins at address 65486 ?

```

main()
{
    intarr[] = {12,14,15,23,45};
    printf("%u %u",arr, &arr);
}

```

4. Experiment the expressions arr and &arr gives same for an array of 10 integers ? (arr is the array variable

- name)
5. Test for the output of the following program, if the array begins at address 65486 ?

```
main()
{
    intarr[] = {12,14,15,23,45};
    printf("%u %u",arr + 1, &arr + 1);
}
```

6. Test for the output of the following program ?

```
main()
{
    float a[] = {12.4,2.3,4.5,6.7};
    printf("\n %d",sizeof(a) / sizeof(a[0]));
}
```

7. Test for the output of the following program if the array begins at 65472?

```
main()
{
    int a[3][4] = {
        1,2,3,4,
        4,3,2,1,
        7,8,9,0
    };
    printf("\n %u %u",a + 1, &a + 1);
}
```

8. Inspect if we pass the name of a 1-D int array to a function it decays into a pointer to an int. If we pass the name of a 2-D array of integers to a function what would it decay into?

#### Multiple Choice Questions

(All MCQ's belongs to (CO3,4,5))

1. An array \_\_\_\_\_ (BL1)
  - a) Is collection of similar data items
  - b) stored data in the continuous memory allocation
  - c) elements share a common name
  - d) All of the above
2. What will be the values of 8<sup>th</sup> element in the array if this is the declaration and initialization of the integer array int a[10]={0};(BL1)
  - a) 0
  - b) Garbage Value
  - c) Value depends on compiler
  - d) Compilation error
3. Pick the correct declaration and initialization of 2 dimensional integer array(BL1)

- a) int a[][]={1,2,2,4};  
 b) int a[3][] ={1,2,2,4};;  
 c) int []a[]={1,2,2,4};  
**d) int a[][3]={1,2,2,4};**
4. Index of the first element in the arrays is \_\_\_\_\_(BL1)
- a) 0  
 b) -1  
 c) 1  
 d) 2
5. printf("%d",a[3][4]); In this statement which element is accessed?(BL1)
- a) 4<sup>th</sup> element in the 3<sup>rd</sup> row of the matrix  
 b) 3<sup>rd</sup> element in the 4<sup>th</sup> row of the matrix  
**c) 5<sup>th</sup> element in the 4<sup>th</sup> row of the matrix**  
 d) 4<sup>th</sup> element in the 5<sup>th</sup> row of the matrix
6. Elements in an array can be accessed\_\_\_\_(BL2)
- a) Sequentially  
 b) Randomly  
**c) Both**  
 d) None  
 e)
7. We want to access the 8<sup>th</sup> element in a one dimensional array int a[50] . which statement is correct?
- a) printf("%d", a[8]);  
**b) printf("%d", a[7]);**  
 c) printf("%d", a[6]);  
 d) printf("%d", a[5]);
8. In the binary search if the key exist in the middle of the array. How many comparison need to be done.  
(BL2)
- a) 1  
 b) n/2  
 c) n  
 d) 0
9. In the linear search if the key exist in the middle of the array. How many comparison need to be done.  
(BL2)
- a) 1  
**b) n/2**  
 c) n  
 d) 0
10. Consider a two dimensional array int a[5][5]. Assume each element occupies 2 bytes. The base address

- of the array is 1000, elements are stores in row major order. What is the address of a[3][2]?
- 1034(BL3)**
  - 1030
  - 1036
  - 1032
11. The maximum dimensional array variable that can be declared in C. (BL2)
- 1
  - 2
  - 3
  - No limit**
12. Pick the operations which cannot be done on an array variable 'a' (BL2)
- $++a$ ,  $a++$
  - $a+1$ ,  $1+a$ ,  $a=a+1$ ,  $a+=1$
  - $a[i]++$ ,  $i[a]++$ ,  $*(a+i)++$ ,  $*(i+a)++$ 
    - both a and b are not allowed**
    - both a and c are not allowed
    - both c and b are not allowed
    - none
13. Pick the uses of arrays (BL2)
- No need to declare many separate variable to handle many data items
  - Using single variable and subscript all the data items can be accessed
  - Maintaining the code is easy and remembering of variables is reduced on programmers side
  - All**
14. Pick the uses of multidimensional arrays (BL2)
- Able to store data in rows and columns
  - It is an array of arrays
  - In all the dimensional the index will start from '0' and ends at 'sizeofthedimension-1'
  - All**
15. In CBIT in the academic year 2020-2021 first year contains 20 sections, in each section 60 students exist, each student is having 6 courses. The secured by all students in all subjects in 20 sections need to be stored. Pick the suitable multidimensional variable to be used to store the data. If data to be stored in row major order of sections, students and subjects (BL3)
- Marks[20][60][6];**
  - Marks[60][20][6];
  - Marks[20][6][60];
  - Marks[6][60][20];
- Review and revision exercise (2) :**
- Write a program to find the determinant of a matrix. (CO3,4,5)
  - Write a program to count number of duplicate elements exist in an array.(CO3,4,5)
  - Write a program to display each element of the array in words.(CO3,4,5)
  - Write a program to display array elements in pyramid form.(CO3,4,5)

Example: 1 2 3 4 5 6 are array elements. Then output should be

- |   |   |   |
|---|---|---|
| 1 |   |   |
| 2 | 3 |   |
| 4 | 5 | 6 |
5. Write a program to find the inverse of a square matrix.(CO3,4,5)
  6. Write a program to perform the below operations on elements of a matrix(CO3,4,5)
    - Find the minimum and maximum element
    - Sort them in ascending order and store in another one dimensional variable
    - Count how many zeros exist in the matrix
    - Count how many duplicates exist
    - Swap any two elements in the matrix based on user choice

#### **Discussion questions / Expected Questions:**

##### **Questions under Blooms Level-1**

1. List out the advantages of an array. (CO5)
2. Define an array. Declare an array which holds average of 60 students.(CO5)
3. What is the significance of index in an array. (CO5)
4. int a[n]; does this declaration is correct? Justify your answer. (CO5)
5. Outline the significance of array name. (CO5)
6. What is the significance of type and size in the array declaration? (CO5)

##### **Questions under Blooms Level-6**

7. Develop a program for sorting n integers using selection sort/bubble sort. (CO3,4,5)
8. Distinguish Lvalue and Rvalue of an array element. Explain the difference with example. (CO5)
9. Develop a program to store 'n' terms of a Fibonacci series in a variable length array. (CO3,4,5)
10. Discuss binary search/linear search with appropriate example in detail. (CO5)
11. Discuss bubble sort/selection sort with appropriate example in detail. (CO5)

##### **Questions under Blooms Level-2,3**

12. Explain in detail about the initialization, access and memory allocation of one dimensional arrays with the help of a diagram. (CO5)
13. Explain in detail about the initialization, access and memory allocation of two dimensional arrays with the help of a diagram. (CO5)
14. Compare a scalar variable and arrays. (CO5)
15. Demonstrate with an example about passing an array to a function.(CO3,4,5)
16. Illustrate with an example about variable length array.(CO5)
17. An integer array is declared of size 10. If the first element of the array is stored at 0x12345 memory location. Find the memory location where 6<sup>th</sup> element of the array will be. (CO5)



**CHAITANYA BHARATHI  
INSTITUTE OF TECHNOLOGY (A)**

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. [www.cbit.ac.in](http://www.cbit.ac.in)

Admitted to



ISO  
9001:2015

COMMITTED TO  
RESEARCH,  
INNOVATION AND  
EDUCATION

**41**  
years

**Course Code:** 20CS C01

**Course Title:** PROGRAMMING FOR PROBLEM SOLVING

**Objective:**

The primary goals of the module: Strings are to introduce:

- Basic representation of Strings and to learn about one-dimensional strings and the way they are declared, initialized, manipulated, inputted, and displayed.
- To know about array of strings, its declaration, initialization, other operations, manipulations, and uses.

**Outcome:**

On Successful completion of the module: Strings, students will be able to:

- learn about one-dimensional and two-dimensional strings and the way they are declared, initialized, manipulated, inputted, and displayed.

**Scope:**

The Module covers the following topics: Introduction to Strings, Strings representation, String operations with examples, Case Study on arrays

While the topics are taught using a C language, the intent of the course is to teach a programming methodology and also a laboratory component that involves development and testing of iterative and procedural programs using one-dimensional and two-dimensional Strings.

**Text and Reference:****(a) Text Book:**

Pradeep Dey and Manas Ghosh, "Programming in C", Oxford Press, 2<sup>nd</sup> Edition, 2017

**(b) Reference Books:**

E. Balaguruswamy, Programming in ANSI C, Tata McGraw-Hill.

Module/ Lecture	Theme	Learning Objectives
L.24	Introduction to Strings, Strings representation.	To learn basic representation of Strings and to learn about one-dimensional strings and the way they are declared, initialized, manipulated, inputted, and displayed.
L.25	String operations with examples.	To know about array of strings, its declaration, initialization, other operations, manipulations, and uses.

L.26	Case Study on arrays and Discussion of Previous Question Papers.	The general purpose of a case study is to: → describe an individual situation (case), in detail; → identify the key issues of the case; → analyse the case using relevant theoretical concepts from the module; → recommend a course of action for that particular case .
------	--	---

<b>Lecture title</b>	L.24 to L.26
----------------------	--------------

<b>Date</b>	
-------------	--

**Key topics :** StringsIntroduction, strings representation, string operations with examples.  
 Case study using arrays.

## STRINGS: ONE-DIMENSIONAL CHARACTER ARRAYS

### Definition:

Strings are defined as an array of characters. The difference between a character array and a string is the string is terminated with a special character ‘\0’ the *null character*, which is a character all of whose bits are zero, i.e., a NUL (not a NULL). (The null character has no relation except in name to the *null pointer*. In the ASCII character set, the null character is named NUL.)

Although C does not have a string data type, it allows string constants. For example: "hello students" is a string constant.

### Declaration of a String:

Strings can be declared like one-dimensional arrays. For example,

```
char str[30];
char text[80];
```

### String Initialization:

Character arrays or strings allow a shorthand initialization, for example,

```
char str[9] = "I like C";
```

which is the same as

```
char str[9] = {‘I’, ‘ ', ‘l’, ‘i’, ‘k’, ‘e’, ‘ ', ‘C’, ‘\0’};
```

Whenever a string, enclosed in double quotes, is written,C automatically creates an array of characters containing that string, terminated by the \0 character. C language allows the alternative notation

```
char msg[] = "Hello";
```

that is always used in practice. It should be noted that the size of the aggregate ‘msg’ is six bytes,

five for the letters and one for the terminating NUL. There is one special case where the null character is

not automatically appended to the array. This is when the array size is explicitly specified and the number of initializers completely fills the array size. For example, `char c[4] = "abcd";` Here, the array `c` holds only the four specified characters, `a`, `b`, `c`, and `d`. No null character terminates the array.

### String Representation:

Index	0	1	2	3	4	5
Variable	H	e			o	\0
Address	0x23451	0x23452	0x23453	0x23454	0x23455	0x23456

### String Input and Output:

#### Printing Strings:

The conversion type ‘s’ may be used for output of strings using `printf()`. Width and precision specifications may be used with the `%s` conversion specifier. The width specifies the minimum output field width; if the string is shorter, then space padding is generated. The precision specifies the maximum number of characters to display. If the string is too long, it is truncated. A negative width implies left

justification of short strings rather than the default right justification. For example, `printf("%7.3s",name);`

This specifies that only the first three characters have to be printed in a total field width of seven characters and right justified in the allocated width by default. We can include a minus sign to make it left justified (`%-7.3`). The following points should be noted:

- When the field width is less than the length of the string, the entire string is printed.
- The integer value on the right side of the decimal point specifies the number of characters to be printed.
- When the number of characters to be printed is specified as zero, nothing is printed.
- The minus sign in the specification causes the string to be printed as left justified.

Example program:

```
#include <stdio.h>
int main()
{
    char s[]="Hello, World";
    printf(">>%s<<\n",s);
    printf(">>%20s<<\n",s);
    printf(">>%-20s<<\n",s);
    printf(">>.4s<<\n",s);
    printf(">>%-20.4s<<\n",s);
    printf(">>%20.4s<<\n",s);
```

```

return 0;
}
Output:
>>Hello, World<<
>>      Hello, World<<
>>Hello, WorlId      <<
>>Hell<<
>>Hell           <<
>>          Hell<<

```

The gets() and puts() are declared in the header file stdio.h. Both the functions are involved in the input/output operations of the strings.

### **Using %s control string with scanf()**

Strings may be read by using the %s conversion with the function scanf() but there are some irksome restrictions. The first is that scanf() only recognizes a sequence of characters delimited by white space characters as an external string. The second is that it is the programmer's responsibility to ensure that there is enough space to receive and store the incoming string along with the terminating null which is automatically generated and stored by scanf() as part of the %s conversion. The associated parameter in the value list must be the address of the first location in an area of memory set aside to store the incoming string.

Of course, a field width may be specified and this is the maximum number of characters that are read in, but remember that any extra characters are left unconsumed in the input buffer. A simple use of scanf() with %s conversions is illustrated in the following program.

```

int main()
{
char str[50];
printf("Enter a string");
scanf("%s",str);
printf("The string was :%s\n",str);
return 0;
}

```

Output of sample runs:

- (a) Enter a string manas  
The string was :manas
- (b) Enter a string manasghosh  
The string was :manas
- (c) Enter a string manas and ghosh  
The string was : “manas”

### C gets() function

The gets() function enables the user to enter some characters followed by the enter key. All the characters entered by the user get stored in a character array. The null character is added to the array to make it a string. The gets() allows the user to enter the space-separated strings. It returns

the string entered by the user.

```
char[] gets(char[]);
```

The gets() function is risky to use since it doesn't perform any array bound checking and keep reading the characters until the new line (enter) is encountered. It suffers from buffer overflow

Example:

```
#include<stdio.h>
void main ()
{
    char s[30];
    printf("Enter the string? ");
    gets(s);
    printf("You entered %s",s);
}
```

C puts() function

The puts() function is very much similar to printf() function. The puts() function is used to print the string on the console which is previously read by using gets() or scanf() function. The puts() function returns an integer value representing the number of characters being printed on the console. Since, it prints an additional newline character with the string, which moves the cursor to the new line on the console, the integer value returned by puts() will always be equal to the number of characters present in the string plus 1

```
int puts(char[]);
```

Example:

```
#include<stdio.h>
#include <string.h>
int main(){
    char name[50];
    printf("Enter your name: ");
    gets(name); //reads string from user
    printf("Your name is: ");
    puts(name); //displays string
    return 0;
}
```

### Character Manipulation in the String:

Character functions in <ctype.h> where c is the character argument.

Function	Description
isalnum(c)	Returns a non-zero if c is alphabetic or numeric
isalpha(c)	Returns a non-zero if c is alphabetic
scntrl(c)	Returns a non-zero if c is a control character
isdigit(c)	Returns a non-zero if c is a digit, 0 – 9
isgraph(c)	Returns a non-zero if c is a non-blank but printing character
islower(c)	Returns a non-zero if c is a lowercase alphabetic character, i.e., a – z
isprint(c)	Returns a non-zero if c is printable, non-blanks and white space included
ispunct(c)	Returns a non-zero if c is a printable character, but not alpha, numeric, or blank
isspace(c)	Returns a non-zero for blanks and these escape sequences: '\f', '\n', '\r', '\t', and '\v'
isupper(c)	Returns a non-zero if c is a capital letter, i.e., A – Z
isxdigit(c)	Returns a non-zero if c is a hexadecimal character: 0 – 9, a – f, or A – F
tolower(c)	Returns the lowercase version if c is a capital letter; otherwise returns c
toupper(c)	Returns the capital letter version if c is a lowercase character; otherwise returns c

Example:

This program checks whether character is alphanumeric or not.

```
/* C example program of isalnum().*/
#include<stdio.h>
#include<ctype.h>

int main()
{
    char ch;

    printf("Enter a character: ");
    scanf("%c",&ch);

    if(isalnum(ch))
        printf("%c is an alphanumeric character.\n",ch);
    else
        printf("%c is not an alphanumeric character.\n",ch);
```

```

        return 0;
    }

```

### String Manipulation:

The string header, string.h, provides many functions useful for manipulating strings or character arrays. Some of these are mentioned below:

Function	Description
<code>strcpy(s1,s2)</code>	Copies s2 into s1
<code>strcat(s1,s2)</code>	Concatenates s2 to s1. That is, it appends the string contained by s2 to the end of the string pointed to by s1. The terminating null character of s1 is overwritten. Copying stops once the terminating null character of s2 is copied.
<code>strncat(s1,s2,n)</code>	Appends the string pointed to by s2 to the end of the string pointed to by s1 up to n characters long. The terminating null character of s1 is overwritten. Copying stops once n characters are copied or the terminating null character of s2 is copied. A terminating null character is always appended to s1.
<code>strlen(s1)</code>	Returns the length of s1. That is, it returns the number of characters in the string without the terminating null character.
<code>strcmp(s1,s2)</code>	Returns 0 if s1 and s2 are the same Returns less than 0 if s1 < s2 Returns greater than 0 if s1 > s2
<code>strchr(s1,ch)</code>	Returns pointer to first occurrence ch in s1
<code>strstr(s1,s2)</code>	Returns pointer to first occurrence s2 in s1

### Example programs:

The given code shows the use of the strcpy() function.

```

#include <string.h>
int main()
{
    char s1[] = "Hello, world!";
    char s2[20];
    strcpy(s2, s1);
    puts(s2);
    return 0;
}

```

The following program illustrates the comparison of two strings.

```

#include <stdio.h>
#include <string.h>

```

```

int main()
{
    char x[50],y[]="a programming example";
    strcpy(x,"A Programming Example");
    if(strcmp(x,"A Programming Example") == 0)
        printf("Equal \n");
    else
        printf("Unequal \n");
    if( strcmp(y,x) == 0)
        printf("Equal \n");
    else
        printf("Unequal \n");
    return 0;
}

```

The following program illustrates the strings concatenation.

```

int main()
{
    char s[30] = "Hello,";
    char str[] = "world!";
    printf("%s\n", s);
    strcat(s, str);
    printf("%s\n", s);
    return 0;
}

```

## ARRAYS OF STRINGS: TWO-DIMENSIONAL CHARACTER ARRAY

### **Declaration:**

A two-dimensional array of strings can be declared as follows:

<data\_type><string\_array\_name>[<row\_size>][<columns\_size>];

Consider the following example on declaration of a two-dimensional array of strings.  
char s[5][30];

### **Initialization:**

Two-dimensional string arrays can be initialized as shown

char s[5][10] = {"Cow", "Goat", "Ram", "Dog", "Cat"};  
which is equivalent to

<b>s[0]</b>	c	o	w	\0					
<b>s[1]</b>	g	o	a	t	\0				
<b>s[2]</b>	r	a	m	\0					
<b>s[3]</b>	d	o	g	\0					
<b>s[4]</b>	c	a	t	\0					

Here every row is a string. That is, s[i] is a string.

Note that the following declarations are invalid.

```
char s[5][] = {"Cow", "Goat", "Ram", "Dog", "Cat"};
char s[][] = {"Cow", "Goat", "Ram", "Dog", "Cat"};
```

Examples:

The following program demonstrates how an individual string of an array of strings can be used to take input from the user. As mentioned before, each row (i.e., s[i], if 's' is the array of strings) of an array of strings is a string.

```
#include <stdio.h>
int main()
{
inti;
char s[10][30], t[30];
for(i=0;i<10;i++)
scanf("%s",s[i]);
for(i=0;i<10;i++)
printf("\n%s",s[i]);
return 0;
}
```

The following program sorts an array of strings using bubble sort. Note here that strcmp() is used to compare the string. strcpy() is used for interchanging the strings.

```
#include <stdio.h>
#include <string.h>
int main()
{
inti,j,n;
char s[10][30], t[30];
inti,j,n;
printf("\n how many strings:");
scanf("%d",&n);
printf("\n enter the strings:\n");
for(i=0;i<n;i++)
scanf("%s",s[i]);
printf("\n **starting comparing and sorting**");
for(i=0;i<n-1;i++)
```

```

for(j=i+1; j<n; ++j)
if(strcmp(s[i],s[j])>0)
{
strcpy(t,s[i]);
strcpy(s[i],s[j]);
strcpy(s[j],t);
}
printf("\n **sorted array**\n");
for(i=0;i<n;i++)
printf("\n%s",s[i]);
return 0;
}

```

### **Summary:**

- Strings are an array of characters terminated by ‘\0’.
- Character arrays or strings allow a shorthand initialization.
- Although C does not have a string data type, it allows string constants.
- There are a set of input and output functions in C suitable for handling strings.
- The manipulation of strings can be carried out with the help of several functions provided in the <string.h> file.
- Arrays can also be formed with strings. These are categorized as two-dimensional arrays.

### **Review and revision:**

#### **MCQs:**

1. Which function will you choose to join two words?(L1)(CO4)
  - a) strcpy()
  - b) strcat()
  - c) strncon()
  - d) memcon()

Answer: b

2. Show the function that appends not more than n characters.(L2)(CO4)
  - a) strcat()
  - b) strcon()
  - c) strncat()
  - d) memcat()

Answer: c

3. Which of the following is the variable type defined in header <string.h>?(L1)(CO3)
  - a) size\_t
  - b) size
  - c) size\_t
  - d) size-t

Answer: c

4. Identify whether NULL is the macro defined in the header <string.h>.(L3)(CO5)
- true
  - false
- Answer: a
5. Recall is there any function declared as strstr()?(L1)(CO4)
- true
  - false
- Answer: a
6. Analyze which of the following function returns a pointer to the located string or a null pointer if string is not found.(L4)(CO4)
- strtok()
  - strstr()
  - strspn()
  - strrchr()
- Answer: b
7. Identify which of the given function is used to return a pointer to the located character?(L3)(CO4)
- strrchr()
  - strxfrm()
  - memchar()
  - strchr()
- Answer: d
8. Analyze what will be the output of the following C code?(L4)(CO5)
- ```
char str1[] = "Helloworld ";
char str2[] = "Hello";
intlen = strspn(str1, str2);
printf("Length of initial segment matching %d\n", len );
```
- 6
  - 5
  - 4
  - no match
- Answer: b
9. Identify thefunction that returns the number of characters that are present before the terminating null character.(L3)(CO4)
- strength()
  - strlen()
  - strlent()
  - strchr()
- Answer: b
10. Examine the output of the following C code?(L4)(CO5)
- ```
char str1[15];
char str2[15];
int mat;
strcpy(str1, "abcdef");
strcpy(str2, "ABCDEF");
mat= strncmp(str1, str2, 4);
```

```

if(mat< 0)
printf("str1 is not greater than str2");
else if(mat> 0)
printf("str2 is not greater than str1");
else
printf("both are equal");

```

- a) str1 is not greater than str2
- b) str2 is not greater than str1
- c) both are equal
- d) error in given code

Answer: b

11. Which function tests for any character for which isalpha or isdigit is true.(L1)(CO4)

- a) isxdigit()
- b) isspace()
- c) isalnum()
- d) isupper()

Answer: c

12. Which function returns true only for the characters defined as lowercase letters?(L1)(CO4)

- a) islow()
- b) islower()
- c) isalpa()
- d) isalnum()

Answer: b

### **Review and revision exercise (2) :**

#### **Case Study:**

WAP to implement Timetable and Invigilation chart with the following options:

1. Class Timetable generation

(Input: Number of Theory and Lab subjects, No. of faculty and their names, each faculty workload ) theory - 3hrs/week, lab-4hrs/week(split of 2hrs)

2. Faculty Timetable generation

(Input: faculty ID)

3. Faculty free hours available

(Input: faculty ID)

4. Invigilation Chart generation

(Input: No. of days of examination, number of invigilators required per day)

#### **Expected Questions:**

Short answer type:

17. Why do we have a null character ('\0' or NUL) at the end of a string? (L1)(CO5)
18. Apply character manipulation functions and develop a C program that will capitalize all

the letters of a string. (L3,L6)(CO4)

Long answer type:

- 19.** Analyze appropriate string manipulation functions and develop a C program that deletes a word from a sentence. Note that the word may appear any number of times.(L4,L3)(CO4)
- 20.** Develop a C program that reads a line of text and counts all occurrences of a particular word.(L6)(CO5)
- 21.** Explain any four string manipulation functions from <string.h> library with examples. (L2)(CO4)



# CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. [www.cblt.ac.in](http://www.cblt.ac.in)

Established  
in 1979



ISO  
9001:2015

COMMITTED TO  
RESEARCH,  
INNOVATION AND  
EDUCATION

41  
years

**Course Code:** 20CSC01

**Course Title:** Programming for Problem Solving

## Unit IV

### Objective:

(The primary goals of the module)

The primary goals of the module/unit are to introduce:

- Basic representation of pointers and usage of pointers with arrays and functions
- Dynamic memory allocation strategy

### Outcome:

1. On Successful completion of the course, students will be able to Apply arrays, pointers, structures, and unions to solve mathematical and scientific problems.

### Scope:

The Module covers the following topics: Basic pointers, Pointers with Arrays and Dynamic memory allocation.

While the topics are taught using a C language, the intent of the course is to teach the usage of pointers along with arrays to solve large complex problems.

### Text and Reference:

#### (a) Text Book:

**TB1:** M.T. Somashekhar “Problem Solving with C”, 2nd Edition, Prentice Hall India Learning Private Limited 2018.

**TB2:** AKSharma“ComputerFundamentalsand Programming”, 2nd Edition, University Press,2018.

**TB3:** Pradeep Dey and Manas Ghosh, "Programming in C", Oxford Press, 2nd Edition, 2017.

**(b) Reference Books:**

**RB1:** Byron Gottfried, Schaum's "Outline of Programming with C", McGraw-Hill

**RB2:** Brian W. Kernighan and Dennis M. Ritchie, "The C Programming Language", Prentice Hall.

**RB3:** E. Balaguruswamy, "Programming in ANSI C", Tata McGraw-Hill.

Module/ Lecture	Theme	Learning Objectives
L.27	Introduction to Pointers, Pointer declaration, Pointer Variables.	To understand how to define, represent, and how to use Pointers.
L.28	Arrays and Pointers, Pointer Arithmetic and Arrays.	To know various Arithmetic operations that can be performed on Pointers and to understand how to use pointers with arrays to solve complex problems.
L.29	Pointers and strings.	To understand the concept of pointers and strings
L.30	Array of pointers, Dynamic memory allocation.	To know how to use dynamic memory allocation for solving memory allocation problems
L.31	Advantages and drawbacks of pointers	To understand pros and cons of pointers

Lecture title	L27 to L31
Date	
Key topics / themes of the lecture :	
<ul style="list-style-type: none"><li>• Pointers</li><li>• Arrays and Pointers</li><li>• Dynamic Memory Allocation</li></ul>	
Notes, comments and your own Examples with answers / questions	
Pointer :	

A pointer is a variable that stores the address of another variable. Unlike other variables that hold values of a certain type, pointer holds the address of a variable. For example, an integer variable holds (or you can say stores) an integer value, however an integer pointer holds the address of a integer variable.

### **Definitions:**

**Pass by address:** facilitating the changes made to a variable in the called function to become permanently available in the function from where the function is called.

**Pass-by-value :**A particular way of implementing a function call, in which the arguments are passed by their value (i.e., their copies).

**Dangling pointer:** A pointer pointing to a previously meaningful location that is no longer meaningful; usually a result of a pointer pointing to an object that is deallocated without resetting the value of the pointer.

**Dynamic memory allocation :**The process of requesting and obtaining additional memory segments during the execution of a program.

**Function pointer:** A function has a physical location in memory that can be assigned to a pointer. Then it is called function pointer. This address is the entry point of the function and it is the address used when the function is called.

**Garbage collection:** If only implicit dynamic allocation is allowed then deallocation must also be done by implicit means, which is often called garbage collection.

**NULL :** A special C constant, defined as macro in stdio.h as 0 or (void\*), that can be used as the null value for pointers.

**null pointer:** A null pointer is a special pointer value that points nowhere. It is initialized with value 0 or NULL.

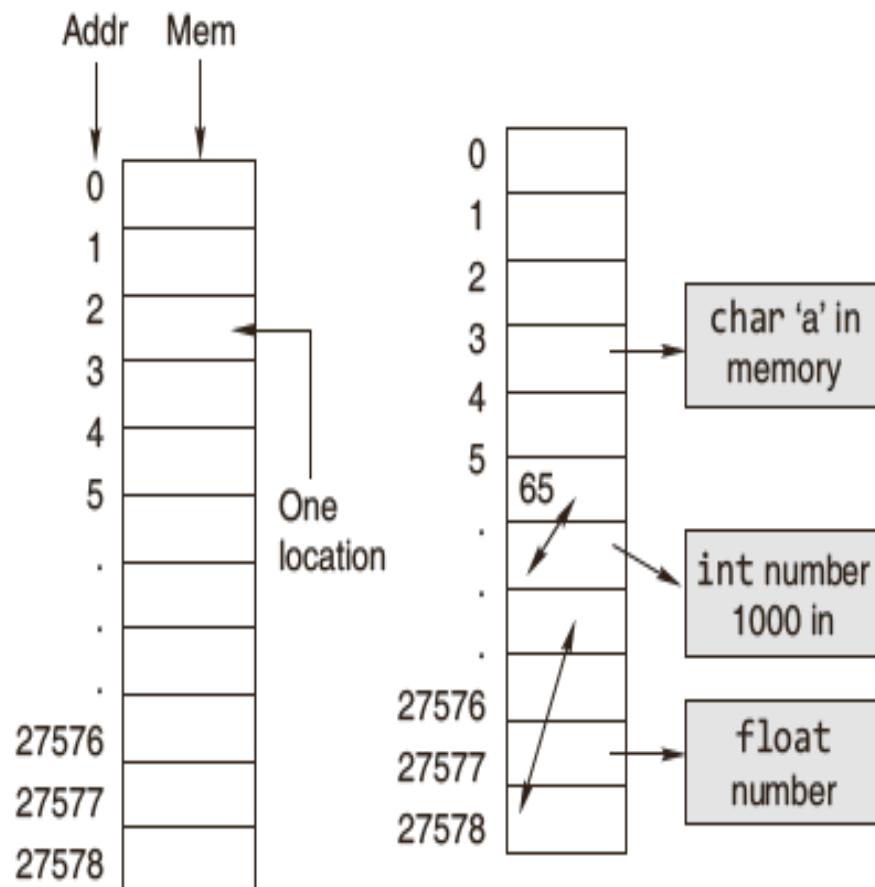
**inter :** A value or a variable with two attributes:

- an address and
- a data type of what should be found at that address.

**static memory allocation:** Memory layout for static data prepared by the compiler.

**void pointer :** A void pointer is a special type of pointer that can point to any data type.

#### Understanding Memory Addresses :



**Figure** The computer memory (16-bit system)

## **Memory Layout:**

1. **Text/Code segment:** code and read-only, sharable by several instances of processes
2. **Data Segment:** Storage for initialized global variables
3. **BSS:** storage for uninitialized global variables
4. **Stack Segment**
  - Provides storage for local variables declared in functions
  - Stores housekeeping information involved when function calls are made
  - Needed for recursive calls
5. **Heap:** Reserved for dynamically(run-time) allocating memory for variables
6. **Shared libraries:** Contains the executable libraries being used by the program

## **Operators used with Pointer:**

### **1. Address Operator (&):**

- Attributes of variables
  - Name
  - Location (address)
  - Type and value
  - Scope, lifetime
  - Operations
- The location of a variable is decided by the compiler and the operating system at run-time.
- The initial value of a variable without assigning a value, the result is a garbage value.
- The ampersand (&) operator is used to get the address of a variable

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

**i**

### **Example:**

- the declaration `int i=3;`
- tells the C compiler to
- reserve space in memory to hold the integer value
  - associate the name `i` with this memory location
  - store the value 3 at this location



**2147478276**

## **2. Indirection Operator and Dereferencing :**

- The primary use of a pointer is to access and, if appropriate, change the value of the variable that the pointer is pointing to.
- The other pointer operator available in C is ‘\*’, called the ‘value at address’ operator. It returns the value stored at a particular address. It is also called indirection operator/ dereference operator.

### **Pointer Variable :**

#### **Definition :**

A pointer is a variable whose value is the address of another variable, i.e., direct address of the memory location. Like any variable or constant, you must declare a pointer before using it to store any variable address.

The general form of a pointer variable declaration is

**Syntax : type \*var-name;**

- A pointer provides a way of accessing a variable without referring to the variable directly.
- A pointer is a variable that holds the memory address of another variable.
- A pointer has two parts,
  - The address that it holds and
  - The value pointed by that address

#### **Pointers can be used to:**

- Call by address, thereby facilitating the changes made to a variable in the called function to become permanently available in the function from where the function is called.
- Return more than one value from a function indirectly.
- Pass arrays and strings more conveniently from one function to another.
- Manipulate arrays more easily by moving pointers to them (or to parts of them) instead of moving the arrays themselves.
- Create complex data structures, such as linked lists and binary trees, where one data structure must contain references to other data structures.
- Communicate information about memory, as in the function **`malloc()`** which returns the location of free memory by using a pointer.
- Compile faster, more efficient code than other derived data types such as arrays.

### **Declaring a Pointer – Pointer Declaration :**

- Like other variables, in a program, a pointer has to be declared.
- A pointer will have a
  - value, scope, lifetime, and name.
- Pointers will occupy a certain number of memory locations.
- The pointer operator available in C is ‘\*’, called ‘value at address’ operator.
- The syntax for declaring a pointer variable is  
**datatype \* pointer\_variable**
- The data type of pointer and the variable must match, an int pointer can hold the address of int variable, similarly a pointer declared with float data type can hold the address of a float variable.

Declaration	What it means
int p	p is an integer
int *p	p is a pointer to an integer
char p	p is a character
char *p	p is a pointer to a character
long p	p is a long integer
long *p	p is pointer to a long integer
unsigned char p	P is an unsigned character
unsigned char *p	P is pointer to an unsigned character

### **Some Pointer Variable Declarations**

#### **Example:**

```
/* Program to illustrate Pointer Variable Declaration*/
#include<stdio.h>

int main( )
{
    int *p;
    float *q;
    double *r;
    printf("\n the size of integer pointer is %d", sizeof(p));
    printf("\n the size of float pointer is %d", sizeof(q));
    printf("\n the size of double pointer is %d", sizeof(r));
    printf("\n the size of character pointer is %d", sizeof(char *));
    return 0;
}
```

**Output :*****In Turbo C:***

the size of integer pointer is 2  
the size of float pointer is 2  
the size of double pointer is 2  
the size of character pointer is 2

***In GCC:***

the size of integer pointer is 4  
the size of float pointer is 4  
the size of double pointer is 4  
the size of character pointer is 4

**Initializing Pointers :**

A Pointer is similar to any other variable excepts that it holds only memory address and therefore it needs to be initialized with a valid address before it can be used.

- Unlike a simple variable that stores a value, a pointer must be initialized with a specified address prior to its use.
- A pointer should be initialized with another variable's memory address, with 0, or with the keyword NULL prior to its use; otherwise the result may be a compiler error or a run-time error.

**Example :**

```
#include <stdio.h>
int main()
{
    int *p; /* a pointer to an integer */
    printf("%d\n", *p);
    return 0;
}
```

**Assigning Pointers :**

- A pointer is bound to a specific data type (except pointer to void) . A pointer to an int cannot

hold the address of a character variable in which case a compiler error would result.

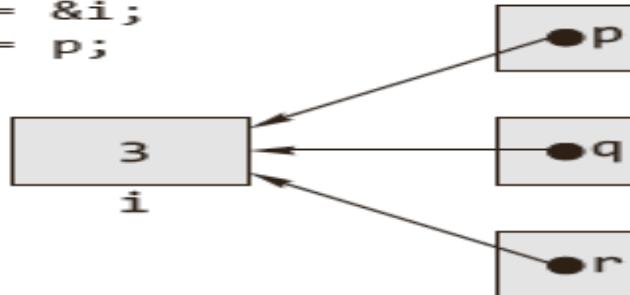
- The following pointer initializations are invalid.

```
int a=6, *ip;  
float *p;  
char ch='A';  
p=&a;  
ip=&ch;
```

From the Below Figure :

- r points to the same address that p points to, which is the address of i.
- We can assign pointers to one another, and the address is copied from the right-hand side to the left-hand side during the assignment.

```
int i=3;  
int *p, *q, *r;  
p = &i;  
q = &i;  
r = p;
```



Printing pointer value :

- To print the memory address stored in pointers and non-pointer variables using the %p conversion specifier and to learn the use of the %p conversion specifier.
- Addresses must always be printed using %u or %p or %x. If %p is used, the address is printed in hexadecimal form. If %u is used, address is printed in decimal form.

Example 1 :

```
/* Program to illustrate Pointer Initialization*/
```

```
#include <stdio.h>  
int main(void)  
{  
    int a=20, *p;  
    *p=&a;  
    printf("\n*p = %p", *p);
```

```
    return 0;  
}
```

**Example 2 :**

```
/* Program to illustrate printing Pointer Value*/
```

```
#include <stdio.h>  
int main(void)  
{  
int a=20, *p;  
p=&a;  
printf("\n*p = %p", *p);  
return 0;  
}
```

**Output:**

```
p = 0022FF2C
```

***Is it possible to assign a constant to a pointer variable?***

- No
- In C, pointers are not allowed to store any arbitrary memory address, but they can only store addresses of variables of a given type.
- Consider the following code:

```
int *pi;  
pi= (int*)1000;  
*pi = 5;
```

- Location 1000 might contain the program. Since it is a read only, the OS will throw up a segmentation fault.

**Example:**

```
/* Program to illustrate printing Pointer Value*/
```

```
#include <stdio.h>  
int main()  
{  
int num = 15;  
int *iPtr = &num;  
printf(" The value of a num is %d", num);  
  
num = 100;
```

```
printf(" The value of a num after num=10 is %d", num);
*iPtr = 25;
printf(" The value of a num after *iPtr = 15 is %d", num);
}
```

#### **Output:**

The value of num is 15  
The value of num after num = 100 is 100  
The value of num after \*iPtr = 25 is 25

#### **Void Pointer :**

- A void pointer is a special type of pointer.
- It can point to any data type, from an integer value or a float to a string of characters.
- Its sole limitation is that the pointed data cannot be referenced directly (the asterisk \* operator cannot be used on them) since its length is always undetermined. Use *type casting* or assignment

#### **Example:**

```
/* Program to illustrate Void Pointer */
```

```
int main()
{
    int a=10,
        double b=3.1415;
    void *vp;
    vp=&a;
    printf("\n a=%d", *((int*)vp));
    vp=&b;
    printf("\n a= %d", *((double *)vp));
    return 0;
}
```

#### **Output :**

a= 10  
b= 3.141500

#### **Null Pointer :**

- Null pointer is a special pointer that points nowhere. That is, no other valid pointer to any other variable or arraycell or anything else will ever be equal to a null pointer.

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

```
int *ip = NULL;
```

- It is also possible to refer to the null pointer using a constant 0,

```
int *ip = 0;
```

- NULL is a constant that is defined in the standard library and is the equivalent of zero for a pointer.

- NULL is a value that is guaranteed not to point to any location in memory.

#### **Use of Pointers :**

**Call by reference :** One of the typical applications of pointers is to support call by reference. However, C does not support call by reference as do other programming languages such as PASCAL and FORTRAN.

- Typically a function call is made to communicate some arguments to the function.
- C makes use of only one mechanism to communicate arguments to a function: **call by value**.
- This means that when a function is called, a copy of the values of the arguments is created and given to the function.
- **Call by reference:** Simulated by the use of pointers called “call by address”.

#### **Example:**

```
/* Program to illustrate call by Value*/
void swap(int a, int b)
{
    int temp; temp=a; a=b; b=temp;
}
int main()
{
    int x=5,y=10;
    void swap(int,int); printf("%d %d\n",x,y);
    swap(x,y);
    printf("%d %d\n",x,y);
    return 0;
}
```

**Output:**

5 10  
5 10                          No swapping

**/\* Program to illustrate call by Address\*/**

```
void swap(int *a, int *b)
{
    int temp; temp = *a;
    *a = *b;
    *b = temp;
}
int main()
{
    int x=5,y=10;
    void swap(int *,int *);
    printf("%d %d\n",x,y);
    swap(&x, &y);
    printf("%d %d\n",x,y); return 0;
}
```

**Output:**

5 10  
10 5

**Returning more than one value from a function:**

- Pointers allow the programmer to return more than one value

ample :

**Program to print area by illustrating returning more than one value from a function\*/**

```
#include <stdio.h>
t main()

    float r, area, perimeter;
    float compute(float, float *); //function prototype
    printf("\n enter the radius of thecircle:");
    scanf("%f",&r);
```

```

area=compute(r, &perimeter); //function call
printf("\n AREA = %f", area);
printf("\n PERIMETER = %f", perimeter); return 0;
}
float compute(float r, float *p)
{
    float a;
    a=(float)3.1415 * r * r;
    *p=(float)3.1415 * 2 * r;
    return a;
}

```

## Arrays and Pointers :

### One-dimensional Arrays and Pointers :

- An array is a non-empty set of sequentially indexed elements having the same type of data.
- Each element of an array has a unique identifying index number. Changes made to one element of an array does not affect the other elements.
- An array occupies a contiguous block of memory. The array a is laid out in memory as a contiguous block, as shown.

The below figure demonstrates the memory allocation strategy of Arrays :

```
int a[]={10, 20, 30, 40, 50};
```

a[0]	a[1]	a[2]	a[3]	a[4]
10	20	30	40	50

2147478270 2147478274 2147478278 2147478282 2147478286

- Array name is an *pointer constant*. It cannot be used as lvalue.
- Array names cannot be used as variables on the left of an assignment operator.
- Both array and &array would give the base address of the array, but the only difference is under ANSI/ISO Standard C, &array yields a pointer, of type pointer to- array of-the data type to the entire array.

**Example :**

```
/* Program to illustrate arrays*/
#include <stdio.h>
int main()
{
int array[]={10, 20, 30, 40, 50};
printf("%u %u", array, &array[0]);
return 0;
}
```

**Output:**

```
2147478270
2147478270
```

*A pointer variable (of the appropriate type) can also be used to initialize or point to the first element of the array.*

```
#include <stdio.h>
int main()
{
    int a[]={10, 20, 30, 40, 50};
    int i, *p;
    p=a; /* it can also be written as p=&a[0]; */
    for(i=0;i<5;i++)
        printf("\n%d", p[i]);

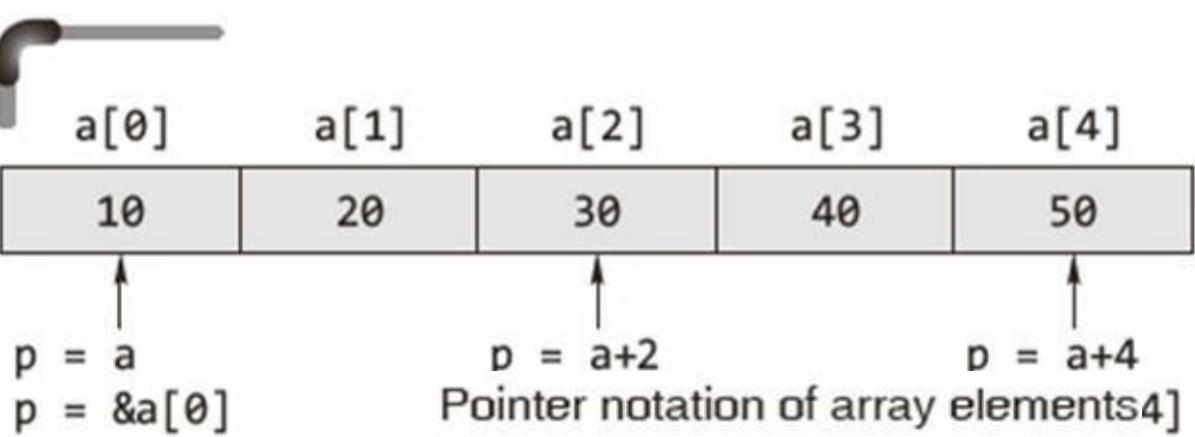
    return 0;
}
```

**Output:**

```
10
20
30
40
50
```

printf ("\n%d", \*(p+i));
OR
printf ("\n%d", \*(i+p));
OR
printf ("\n%d", i[p]);

*Below figure depicts the equivalence among array notation and pointer notation.*



### Passing Array to a Function :

- For any one-dimensional array  $a$  and integer  $i$ , the following relationships are always true.  

$$a[i] \equiv *(a+i) \equiv *(i+a) \equiv i[a]$$
- An array may be passed to a function, and the elements of that array may be modified without having to worry about referencing and dereferencing.

```

int main()
{
    int arr[MAX], n;
    ...
    n = getdata(arr, MAX);
    show(arr, n);
    return 0;
}

int getdata(int *a, int n)
{
    ...
}

void show(int *a, int n)
{
    ...
}

```

`a=arr;` is highlighted in a box and has arrows pointing to its occurrences in `main()` and `getdata()`.

*When an array is passed to a function, it degenerates to a pointer.*

- All array names that are function parameters are always converted into pointers by the compiler.
- Because when passing an array to a function, the address of the zero-th element of the array is copied to the pointer variable which is the formal parameter of the function.
- However, arrays and pointers are processed differently by the compiler, represented differently at runtime.

#### Differences between Array Name & Pointer :

- When memory is allocated for the array, the starting address is fixed, i.e., it cannot be changed during program execution. Therefore, array name is an address constant.
- The & (address of) operator normally returns the address of the operand. However, arrays are the exception.
- When applied to an array (which is an address), it has the same value as the array reference without the operator.
- This is not true of the equivalent pointers, which have an independent address.
- The **sizeof** operator returns the
  - Size of the allocated space for arrays in case of arrays.
  - 4 or 8 bytes depending on the machine architecture.

#### Difference Between Arrays and Pointers :

Arrays	Pointers
Array allocate space automatically	It is explicitly assigned to point to an allocated space
It cannot be resized	It can be resized using realloc( )
It cannot be reassigned	It can be reassigned
sizeof(arrayname) gives the number of bytes occupied by the array	sizeof(p) gives the number of bytes used to store the pointer variable p

#### Pointer Arithmetic :

A pointer in C is an address, which is a numeric value. Therefore, you can perform arithmetic operations on a pointer just as you can on a numeric value. There are four arithmetic operators that can be used on pointers: `++`, `--`, `+`, and `-`

#### Pointer Arithmetic - Valid :

The only valid operations on pointers are as follows.

- 1) Assignment of pointers to the same type of pointers: the assignment of pointers is done symbolically. Hence no integer constant except 0 can be assigned to a pointer.
  - 2) Adding or subtracting a pointer and an integer.
  - 3) Subtracting or comparing two pointers (within array limits) that point to the elements of an array
- Pointer Arithmetic - Valid**
- 4) Incrementing or decrementing the pointers (within array limits) that point to the elements of an array.
    - When a pointer to an integer is incremented by one, the address is incremented by two (as two bytes are used for int).
    - Such scaling factors necessary for the pointer arithmetic are taken care of automatically by the compiler.
  - 5) Assigning the value 0 to the pointer variable and comparing 0 with the pointer. The pointer with address 0 points to nowhere at all.

### **Pointer Arithmetic- not Valid**

1. Addition of two pointers
2. Multiplying a pointer with a number
3. Dividing a pointer with a number

### **Important Points - Pointer Arithmetic:**

Operation	Condition	Example	Result
Assignment	Pointers must be of same type	int *p,*q ...	p points to whatever q points to
Addition of an integer		p = q; int k,*p; ... p + k	Address of the kth object after the one p points to
Subtraction of an integer		int k,*p; ... p - k	Address of the kth object before the one p points to
Comparison of pointers	Pointers pointing to the members of the same array	int *p,*q; ... q < p	Returns true (1) if q points to an earlier element of the array than p does. Return type is int
Subtraction of pointers	Pointers to members of the same array and q < p	int *p,*q; ... p - q	Number of elements between p & q;

### **Pointers and Strings :**

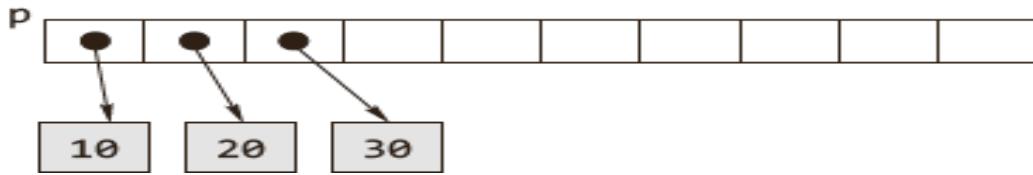
- Strings are one-dimensional arrays of type char. By convention, a string in C is terminated by the end-of-string sentinel \0, or null character.
- A string constant is treated by the compiler as a pointer.
- For a string, the number of elements it needs not be passed to a function because it has the terminating null character

## Array of Pointers :

- An array of pointers can be declared very easily.  
**Example:** `int *p[20];`
- Declares an array of 20 pointers, each of which points to an integer.
- 1<sup>st</sup> pointer is p[0], 2<sup>nd</sup> is p[1], and so on up to p[19].
  - These start off as uninitialized—they point to some unknown point in memory. We could make them point to integer variables in memory thus.

**Example :**

```
int* p[10];
int a = 10, b = 20, c = 30;
p[0] = &a;
p[1] = &b;
p[2] = &c;
```



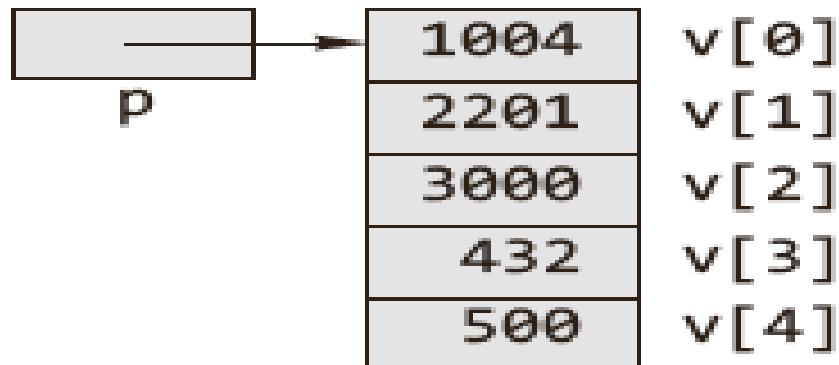
**Pointers to Array :**

We can declare a pointer to a simple integer value and make it point to the array as is done normally.

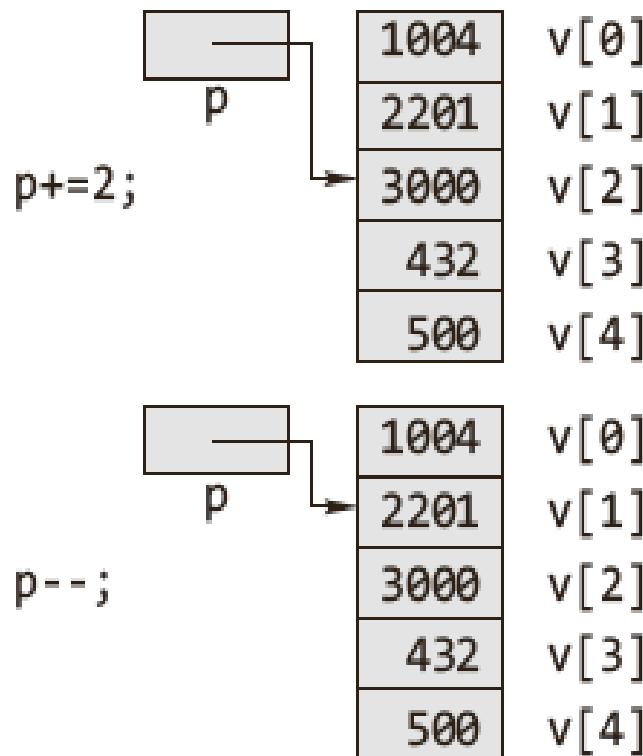
**Example :**

```
int v[5] = {1004, 2201, 3000, 432, 500};
int *p = v;
printf("%d \n", *p);
```

This piece of code displays the number, which the pointer `p` points to, that is the first number in the array, namely 1004.



can use instructions such as `+=` and `-=` to refer to different elements in the array.



### Difference between an array of pointers and a pointer to an array

Array of Pointer	Pointer to an Array
Declaration	Declaration
<code>Data_type * array_name[SIZE];</code>	<code>Data_type (* array_name)[SIZE];</code>
Size represents the number of rows	Size represents the number of columns
The space for columns may be allotted	The space for rows may be dynamically allotted

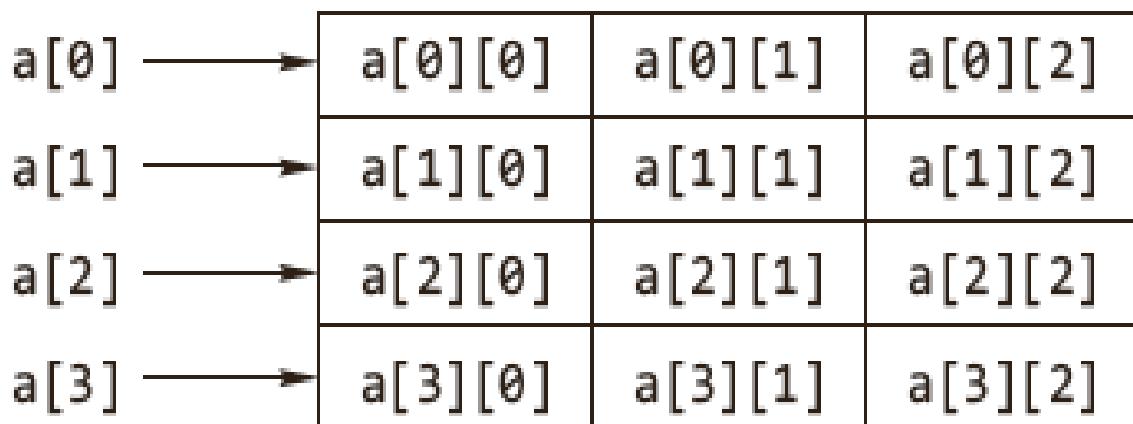
### Two-dimensional array and Pointers :

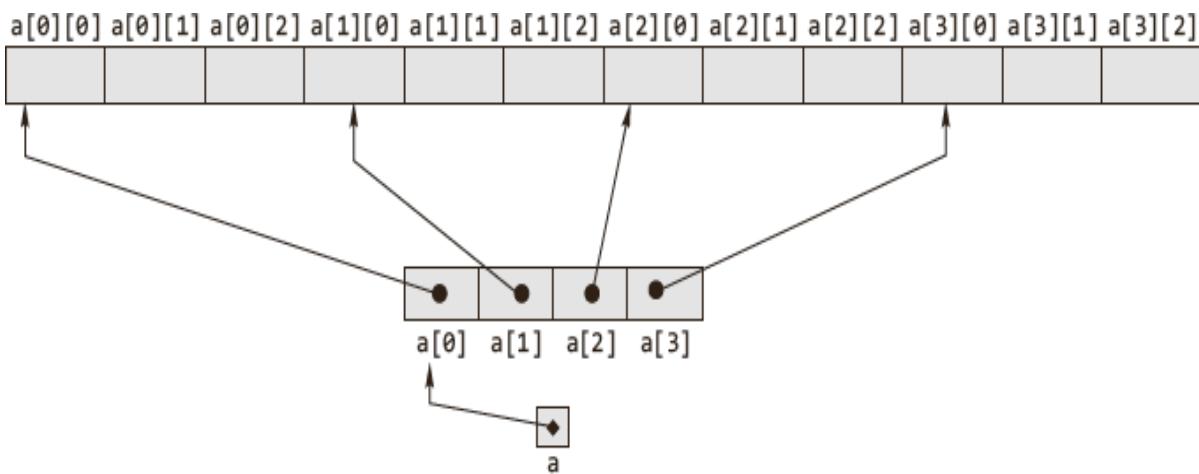
A two-dimensional array in C is treated as a one dimensional array whose elements are one-dimensional arrays (the rows).

- C does not do run-time range checking of array subscripts.
- In C the rightmost subscript of a two-dimensional array varies faster than the leftmost.
- Multi-dimensional array is stored in a 'row major addressing' format.
- The following expressions are equivalent for a two dimensional array
 
$$a[i][j] = *(a[i] + j) = (*a + i)[j] = *(*a + i) + j$$
- Arrays of dimension higher than two work in a similar fashion.
  - Let us describe how three-dimensional arrays work.
  - If the following is declared
 

```
int a[7][9][2];
```
- In case of multi-dimensional arrays all sizes except the first must be specified.

*Logical representation of a two-dimensional array*





Physical representation of a two-dimensional array

### Example 1 :

```
/* Program to illustrate matrix addition – two dimensional array*/

void getdata(int x[10][10], int m, int n)
{
    for(int i=0;i<m;++i) for(int j=0;j<n;++j)
        scanf("%d",&x[i][j]); return;
}
void display(int x[10][10], int m, int n)
{
    for(int i=0;i<m;++i)
    {
        for(int j=0;j<n;++j) printf("%4d",x[i][j]);
        printf("\n");
    }
    return;
}
void add(int x[10][10],int y[10][10],int z[10][10],int m,int n)
{
    for(int i=0;i<m;++i) for(int j=0;j<n;++j)
        z[i][j] = x[i][j] + y[i][j];
    return;
}
```

```

int main()
{
    int A[10][10],B[10][10],C[10][10],T[10][10],m,n;
    printf("\nEnter the values of m and n: ");
    scanf("%d %d",&m,&n);
    printf("\nEnter the elements of A: ");
    getdata(A,m,n);
    printf("\nEnter the elements of B: ");
    getdata(B,p,q);
    printf("\nGiven matrix A is:\n");
    display(A,m,n);
    printf("\nGiven matrix B is:\n");
    display(B,p,q);
    add(A,B,C,m,n);
    printf("\nGiven Resultant matrix C is:\n");
    display(C,m,n);
    return 0;
}

```

**Example 2 :**

```
/* matrix addition – Array of pointers*/
```

```

int *A[10],*B[10],*C[10],m, n, i;
printf("\nEnter the values of m and n: ");
scanf("%d %d",&m,&n);                                //read the values of m and n
for(i=0;i<m;++i)                                     //allocate memory dynamically
{
    A[i]=(int*)malloc(n*sizeof(int));
    B[i]=(int *)malloc(n*sizeof(int));
    C[i]=(int *)malloc(n*sizeof(int));
}
getdata(A,m,n);
getdata(B,m,n);
add(A,B,C,m,n);
display(C,m,n);

```

**Example 3 :**

```

/* matrix addition – Use of pointers*/

void getdata(int *x[10], int m, int n)
{
    for(int i=0;i<m;++i) for(int j=0;j<n;++j)
        scanf("%d",x[i]+j); //or &x[i][j] or *(x+i)+j
    return;
}
void display(int *x[10], int m, int n)
{
    for(int i=0;i<m;++i)
    {
        for(int j=0;j<n;++j)
            printf("%4d",*(x[i]+j)); //or x[i][j] or *(*(x+i)+j)

        printf("\n");
    }
    return;
}

```

**Example 4 :**

```

/* matrix addition – Pointer to 2-D array */

int **A,**B,**C, i, m, n;
printf("\nEnter the values of m and n: "); scanf("%d %d",&m,&n);
                                //read the values of m and n
for(i=0;i<m;++i)
    A[i]=(int *)malloc(n*sizeof(int)); for(i=0;i<m;++i)
    B[i]=(int *)malloc(n*sizeof(int)); for(i=0;i<m;++i)
C[i]=(int *)malloc(n*sizeof(int));

```

**Example 5 :**

```

/* matrix addition – Using Pointers - Pointer to 2-D array */

void getdata(int **x, int m, int n)
{
    for(int i=0;i<m;++i) for(int j=0;j<n;++j)
        scanf("%d", *(x+i)+j); //or &x[i][j] or *(x+i)+j or (x+i)[j]
    return;
}
void display(int *x[10], int m, int n)
{
    for(int i=0;i<m;++i)
    {
        for(int j=0;j<n;++j)
            printf("%4d",*(*(x+i)+j)); //or x[i][j] or *(*(x+i)+j) or *(x+i)[j]

        printf("\n");
    }
    return;
}

```

### **Pointers to Functions :**

C also allows pointers to point to functions and it is one of the powerful feature

Declaration of a Pointer to a Function

return\_type (\*fptr) (argument\_type1, argument\_type2, ..);

**Example:**

```
int(*fp)(int, char, float); //fp is a pointer to a function that take one int, one char, one float
                            and returns an int
```

- Parenthesis are important and `int *fp();` declares a function that returns an integer pointer

### **Initialization of Function Pointers:**

For example , two functions are declared as:

`int add(int, int); and int sub(int, int);`

- The names of these functions, `add` and `sum`, are pointers to those functions. These can be assigned to pointer variables.

`fptr = add;`

```
fptr = sub;
```

### ***Calling a Function using a Function Pointer :***

*For example:*

fptr=add; then the function call via function pointer will be:

result1 = fptr(4, 5); will return 9

fptr=sub;

result2 = fptr(6, 2); will return 4.

### **Passing a Function to another Function:**

function pointer can be passed as a function's calling argument.

- How to Return a Function Pointer

```
float(*GetPtr1(char opCode))(float, float)
{
    if(opCode == '+') return &Add; if(opCode == '-')
        return &Sub;
}
```

A solution using a typedef defines a pointer to a function that takes two float values and returns a float.

### **Arrays of Function Pointers:**

**Example :**

```
/* Program – Arrays of Function Pointers */
```

```
void Add(int a, int b)
{
    printf("\n Result of Addition = %d",a+b);
}

void Sub(int a, int b)
{
    printf("\n Result of Subtraction = %d",a-b);
}

void Mul(int a, int b)
{
    printf("\n Result of Multiplication = %d",a*b);
}

int main()
{
    void(*p[3])(int, int);
```

```
int i;
void Add(int, int);
void Sub(int, int);
void Mul(int, int);
p[0] = Add;
p[1] = Sub;
p[2] = Mul;
for(i = 0; i <= 2; i++)
(*p[i])(10, 5);
return 0;
}
```

## **Dynamic Memory Allocation :**

### **Static memory allocation :**

- Compiler allocates the required memory space for a declared variable.
- By using the address of operator, the reserved address is obtained that maybe assigned to a pointer variable.
- Since most declared variables have static memory, this way of assigning pointer value to a pointer variable is known as **static memory allocation**.

### **Dynamic memory allocation :**

- A dynamic memory allocation uses functions such as malloc() or calloc() realloc() to get memory dynamically.
- If these functions are used to get memory dynamically and the values returned by these functions are assigned to pointer variables, such assignments are known as **dynamic memory allocation**.
- Memory is assigned during run-time.
- In dynamic memory allocation, memory is allocated at runtime from heap. According to ANSI compiler, the malloc() and calloc() returns a void pointer on successful allocation of memory. If sufficient memory is not available, the malloc() and calloc() returns a NULL.
- calloc() initializes all the bits in the allocated space set to zero whereas malloc() does not do this.
- When dynamically allocated, arrays are no longer needed, it is recommended to free them immediately.

### **Dynamic Allocation of Arrays:**

To allocate a one-dimensional array of length N of some particular type where N is given

by the user, simply use **malloc()** to allocate enough memory to hold N elements of the particular type.

**Example :**

```
/* Program – Dynamic memory Allocation */

#include<stdio.h>
#include<stdlib.h>
int main()
{
int N,*a,i,s=0;
printf("\n enter no. of elements of the array:");
scanf("%d",&N);
a=(int *)malloc(N*sizeof(int));
if(a==NULL)
{
printf("\n memory allocation unsuccessful...");
exit(0);
}
printf("\n enter the array elements one by one");
for(i=0; i<N;++i)
{
scanf("%d",&a[i]); /* equivalent statement
                     scanf("%d",*(a+i));*/
s+=a[i];
}
printf("\nsum is%d",s); return 0;
}
```

**Freeing Memory :**

- Dynamically allocated memory is de-allocated with the **free** function. If p contains a pointer previously returned by **malloc**(), a call such as **free(p)**;
- **calloc()** requires **two** parameters, the first for the number of elements to be allocated and the second for the size of each element, whereas **malloc()** requires **one** parameters.
- **calloc()** initializes all the bits in the allocated space set to zero whereas **malloc()** does not do this.
- **P=calloc(m, n)** is essentially equivalent to **p = malloc(m \* n);**
- **malloc(0)** results are unpredictable. It may return some other pointer or it may return some

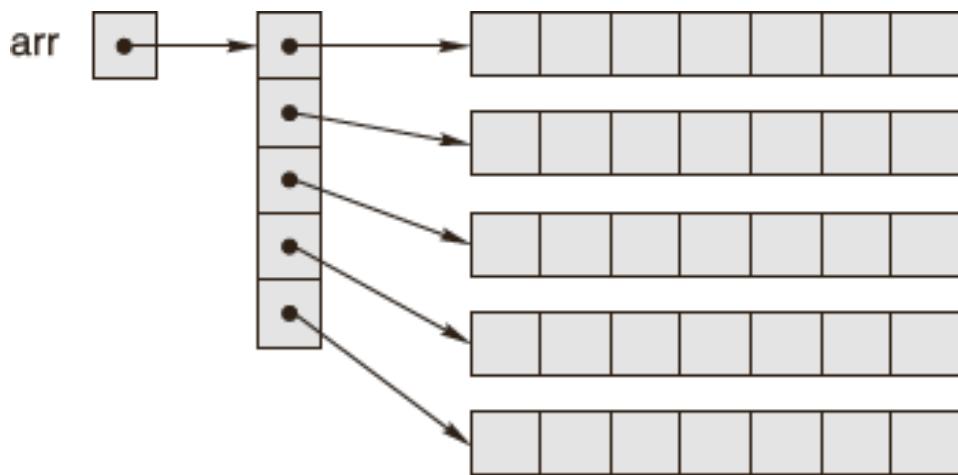
other implementation-dependent value.

### Reallocating Memory Blocks :

- malloc() will not work. It is the realloc() function that is required  
`ip = realloc(ip, 200 * sizeof(int));`

### Implementing Multidimensional Arrays using Pointers:

It is usually best to allocate an array of pointers, and then initialize each pointer to a dynamically allocated 'row'.



### *Memory Leak and Memory Corruption*

#### Dangling pointer :

A pointer may be used to hold the address of dynamically allocated memory. After this memory is freed with the free() function (in C), the pointer itself will still contain the address of the released block. This is referred to as a **dangling pointer**.

#### Memory corruption :

Memory when altered without an explicit assignment due to the inadvertent and unexpected altering of data held in memory or the altering of a pointer to a specific place in memory is known as memory corruption.

#### Advantages of Pointers :

- Pointers save memory space
- Execution time with pointer is fast because data is manipulated with address i.e directly

with memory location.

- Memory is accessed efficiently with pointers
- Dynamic memory is allocated
- Pointers are used with data structures.
- Pointers are useful for representing two dimensional and multi dimensional Arrays.

### **Drawbacks of Pointers :**

- A programmer is likely to commit mistakes such as typographical mistakes and providing wrong offsets.
- Porting the code to other implementations would require changes, if data type sizes differ. This would lead to portability issues.
- Pointers have data types but the size of a pointer variable is always 4-bytes (in a 32-bit machine) 8-bytes (in a 64-bit machine) whatever the data type is used in declaring it.

### **Summary exercise:**

6. [Pointer is a memory variable that holds address of another variable.
7. Certain Arithmetic Operations can be performed on Pointers.
8. An array name itself is a pointer. Whenever address are stored as array elements, such an array is called as array of pointers. 2D arrays can be represented using pointers.
9. Dynamic Memory Allocation is the process by which the required memory address is obtained without an explicit declaration.

### **Review and revision:**

#### **Review and revision exercise (1) :**

#### **Lecture topic quiz / MCQs:**

1. (\*FP)(INT, INT) REPRESENTS .....

L2

**A. FUNCTION POINTER**

**B. ARRAY OF PINTERS**

**C. POINTER ARRAY**

**D. ARRAY OF FUNCTION POINTER**

2. \_\_\_\_\_ IS THE MEMORY VARIABLE USED TO STORE ADDRESS OF ANOTHER VARIABLE.

L1

**A. FUNCTION**

**B. VARIABLE**

**C. POINTER**

**D. IDENTIFIER**

3. INT \*P;

L3

INT A=100;

```
P=&A;  
PRINTF("%d %d",*P,P);
```

**A.100,1000**

B. 10,1000

C. 10

D. 1000

4. Address stored in pointer variable is of \_\_\_\_\_ type

L1

**A. Integer**

B. character

C. Float

D. Double

5. \* is called as \_\_\_\_\_

L1

**A. Value at pointer**

B. Address operator

C. Scope resolution operator

D. None

6. int \*p1,\*p2; find out valid statement

L2

A. p1-p2

**B. p1\*p2**

C. p1+p2

D. p1/p2

7. The operator used to get value at address stored in a pointer variable is..... L1

**A.\***

B.&

C.&&

D. ||

8. What would be the equivalent pointer expression for referring the array element a[i][j][k][l].

L3

A. (((a+i)+j)+k)+l)

**B. \*(\*(\*(a+i)+j)+k)+l)**

C. ((a+i)+j)+k+l)

D. ((a+i)+j+k+l)

9. A pointer is .....

L1

A. A keyword used to create variables

B. A variable that stores address of an instruction

**C. A variable that stores address of other variable**

D. All of the above

10.What will be the output?  
 main()  
 {  
 char \*p;  
 printf("%d %d", sizeof(\*p), sizeof(p));  
 }  
 A.1 1  
**B.1 2**  
 C.2 1  
 D.2 2

L4

**Review and revision exercise (2) :**

**Discussion questions / Expected Questions:**

**Short Questions :**

- |  |    |
|--|----|
| 1. Define pointer. How can you declare it?   | L2 |
| 2. What is pointer arithmetic?               | L2 |
| 3. Define pointer array.                     | L1 |
| 4. What is pointer to pointer?               | L1 |
| 5. Define Void Pointer.                      | L1 |
| 6. Define Null Pointer.                      | L1 |
| 7. Write about operators used with pointers. | L2 |
| 8. Differentiate Pointers and Arrays.        | L3 |

**Long Questions :**

- |  |    |
|--|----|
| 1. Write a program to calculate length of the string using pointers.   | L6 |
| 2. Write a C program to illustrate the use of indirection operator to access the value pointed by a pointer. | L6 |
| 3. What are the features of pointers? Write a C program to print address of a variable.                      | L3 |
| 4. Explain the declaration of pointers and pointer to pointer with examples.                                 | L4 |
| 5. Explain the concept of array of pointers with examples.   | L3 |
| 6. With proper examples explain different arithmetic operations on pointers.                                 | L3 |
| 7. Explain the concept of functions returning pointers with example.   | L3 |
| 8. Write a C program to read and print an array of elements using pointers.                                  | L6 |
| 9. Write a C program to read and display multiple strings using pointers.                                    | L6 |
| 10. Write a C Program to perform matrix addition using dynamic memory allocation.                            | L6 |



**CHAITANYA BHARATHI  
INSTITUTE OF TECHNOLOGY (A)**

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. [www.cbit.ac.in](http://www.cbit.ac.in)



COMMITTED TO  
RESEARCH,  
INNOVATION AND  
EDUCATION

**41**  
years

## UNIT IV

**Course Code:** 20CSC01

**Course Title:** Programming for Problem Solving

### Objective:

(The primary goals of the module/unit)

The primary goals of the module/unit are to introduce:

- Basic representation of Structures and usage of Structures along with functions
- Self Referential structures which helps to understand various data structures in further courses.
- Basic representation of Unions and enumerated datatypes

### Outcome:

1. On Successful completion of the course, students will be able to Apply structures, and unions to solve mathematical and scientific problems.

## **Scope:**

The Module/Unit covers the following topics: Structure definition, initialization and accessing the members of a structure, nested structures, structures and functions, self-referential structures, unions, and enumerated data types.

While the topics are taught using a C language, the intent of the course is to teach the usage of Structures along with functions to solve large complex problems.

## **Text and Reference:**

### **(a) Text Book:**

**TB1:** M.T. Somashekhar “Problem Solving with C”, 2nd Edition, Prentice Hall India Learning Private Limited 2018.

**TB2:** AKSharma “Computer Fundamentals and Programming”, 2nd Edition, University Press, 2018.

**TB3:** Pradeep Dey and Manas Ghosh, “Programming in C”, Oxford Press, 2nd Edition, 2017.

### **(b) Reference Books:**

**RB1:** Byron Gottfried, Schaum’s “Outline of Programming with C”, McGraw-Hill

**RB2:** Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Prentice Hall.

**RB3:** E. Balaguruswamy, Programming in ANSI C, Tata McGraw-Hill .

**RB4:** Reema Tharaja “Introduction to C Programming”, Second Edition, OXFORD Press, 2015.

**RB5:** <https://www.tutorialspoint.com/cprogramming/index.htm>.

<b>Module/ Lecture</b>	<b>Theme</b>	<b>Learning Objectives</b>

L.32	Structures: Definition and Initialization of Structures.	To understand how to define, represent, and how to use Structure.
L.33	Accessing Members of Structures, Nested Structures.	To know how to access members of structures and to understand how to use nested structures.
L.34	Self Referential Structures.	To understand the concept of self Referential Structures
L.35	Structures and Functions.	To know how to pass and return structure as argument in functions
L.36	Unions and Enumerated Types.	To understand how to define, represent, and how to use Unions and Enumerated Types.

<b>Lecture title</b>	( L1 etc)
<b>Date</b>	[ You Can Write During the Course]

### **Key topics / themes of the lecture :**

Structure definition, initialization and accessing the members of a structure, nested structures, structures and functions, self-referential structures, unions, and enumerated data types.

### **Notes, comments and your own Examples with answers / questions**

#### **Definition**

Arrays allow defining type of variables that can hold several data items of the same kind. Similarly **structure** is another user defined data type available in C that allows combining data items of different kinds.

Structures are used to represent a record. Suppose you want to keep track of your books in a library. You might want to track the following attributes about each book—

Title Author Subject Book ID

#### **Defining a Structure**

To define a structure, you must use the **struct** statement. The struct statement defines a new data type, with more than one member. The format of the struct statement is as follows –

```

struct structure_name{ data_type struct_member1; data_type struct_member2;
-----  

    data_type struct_member n;  

} [one or more structure variables separated by comma];

```

The **structure\_name** is optional and each *struct\_member* is a normal variable definition, such as int i; or float f; or any other valid variable definition. At the end of the structure's definition, before the final semicolon, you can specify one or more structure variables but it is optional.

Here is the way you would declare the Book structure –

```

struct Books  

{  

    char title[50]; char author[50]; char subject[100]; int book_id;  

} book;

```

## Accessing Structure Members

To access any member of a structure, we use the **member access operator** (.). The member access operator is coded as a period between the structure variable name and the structure member that we wish to access. You would use the keyword **struct** to define variables of structure type.

The following example shows how to use a structure in a program

```

#include <stdio.h>  

#include <string.h>  

struct Books {  

    char title[50];  

    char author[50];  

    char subject[100];  

    int book_id;  

};  
  

int main()  

{  
  

    Struct Books Book1; /* Declare Book1 of type Book */ struct Books Book2; /* Declare  

Book2 of type Book */  

/* book 1 specification */  

strcpy( Book1.title, "C Programming");  

strcpy( Book1.author, "Nuha Ali");  

strcpy( Book1.subject, "C Programming Tutorial");

```

```

Book1.book_id = 6495407;
/* book 2 specification */
strcpy( Book2.title, "Telecom Billing");
strcpy( Book2.author, "Zara Ali");
strcpy( Book2.subject, "Telecom Billing Tutorial");
Book2.book_id = 6495700;
/* print Book1 info */
printf( "Book 1 title : %s\n", Book1.title);
printf( "Book 1 author : %s\n", Book1.author);
printf( "Book 1 subject : %s\n", Book1.subject);
printf( "Book 1 book_id : %d\n", Book1.book_id);
/* print Book2 info */
printf( "Book 2 title : %s\n", Book2.title);
printf( "Book 2 author : %s\n", Book2.author);
printf( "Book 2 subject : %s\n", Book2.subject);
printf( "Book 2 book_id : %d\n", Book2.book_id); return 0;
}

```

When the above code is compiled and executed, it produces the following result

Book 1 title : C Programming

Book 1 author : Nuha Ali

Book 1 subject : C Programming Tutorial Book 1 book\_id : 6495407

Book 2 title : Telecom Billing

Book 2 author : Zara Ali

Book 2 subject : Telecom Billing Tutorial Book 2 book\_id : 6495700

### Nested structures

Structure written inside another structure is called as nesting of two structures. Nested Structures are allowed in C Programming Language. We can write one Structure inside another structure as member of another structure.

#### **Example:**

```

structdate
{
int date; int month; int year;
};

```

```

struct Employee
{

```

```

char name[20];
int ssn;
float salary;

```

```
struct date DOJ;
}emp1;
```

### Accessing the nested elements

Structure members are accessed using dot operator. ‘date’ structure is nested within Employee Structure. Members of the ‘date’ can be accessed using ‘employee’ emp1 & DOJ are two structure names (Variables)

### Explanation of nested elements:

Accessing Month Field : emp1.Doj.month  
Accessing day Field : emp1.Doj.day  
Accessing year Field : emp1.Doj.year

The following example shows how to use nested structure in a program

```
#include <stdio.h>
struct Employee
{
    char name[20];
    int ssn;
    float salary;
    struct date
    {
        int date;
        int month;
        int year;
    }doj;
}emp = {"Pritesh",1000,1000.50,{22,6,1990}};

int main(int argc, char *argv[])
{
    printf("\nEmployee Name : %s",emp.name);
    printf("\nEmployee SSN      : %d",emp.ssn);
    printf("\nEmployee Salary : %f",emp.salary);
    printf("\nEmployee DOJ      : %d/%d/%d", emp.doj.date,emp.doj.month,emp.doj.year);
    return 0;
}
```

When the above code is compiled and executed, it produces the following result

```
Employee Name :Pritesh Employee SSN : 1000 Employee Salary : 1000.500000 EmployeeDOJ
: 22/6/1990
```

## Arrays of Structures

We may declare the structure variable as an array by using index. Arrays of structures are passed using the following syntax,

```
structstructure_name{ data_type struct_member1; data_typestruct_member2;  
-----  
data_typestruct_membern;  
};  
structstructure_namestruct_variable[index_value];
```

**Example:** #include<stdio.h> #include<conio.h> void main()

```
{  
    struct student  
    {  
        intrno;  
        char name[20]; char course[20]; float fees;  
    };  
    struct student s[20]; clrscr();  
    s[0].rno=10; gets(s[0].name);  
    gets(s[0].course); s[0].fees=1000.50;  
    printf("\n %d\t %s\t %s\t %f",s[0].rno,s[0].name,s[0].course,s[0].fees);  
    getch();  
}
```

## Structures as Function Arguments

You can pass a structure as a function argument in the same way as you pass any other variable or pointer. Like all other types, we can pass structures as arguments to a function. In fact, we can pass individual members, structure variables, a pointer to structures etc to the function. Similarly, functions can return either an individual member or structures variable or pointer to the structure.

## Passing Structure Members as arguments to Function

We can pass individual members to a function just like ordinary variables. The following program demonstrates how to pass structure members as arguments to the function.

```
1 #include<stdio.h>  
2  
3 /*  
4 structure is defined above all functions so it is global.  
5 */  
6  
7 structstudent
```

```

8 {
9 charname[20];
10introll_no;
11intmarks;
12};
13
14voidprint_struct(charname[],introll_no,intmarks);
15
16intmain()
17{
18structstudentstu={"Tim",1,78};
19print_struct(stu.name,stu.roll_no,stu.marks);
20return0;
21}
22
23voidprint_struct(charname[],introll_no,intmarks)
24{
25printf("Name: %s\n",name);
26printf("Roll no: %d\n",roll_no);
27printf("Marks: %d\n",marks);
28printf("\n");
29}

```

When the above code is compiled and executed, it produces the following result

Name: Tim

Roll no: 1

Marks: 78

#### **Flow of Execution of the above code :**

- In lines 7-12, a structure student is declared with three members namely name, roll\_no and marks.
- In line 14, a prototype of function print\_struct() is declared which accepts three arguments namely name of type pointer to char, roll\_no of type int and marks is of type int.
- In line 18, a structure variable stu of type struct student is declared and initialized.
- In line 19, all the three members of structure variable stu are passed to the print\_struct() function. The formal arguments of print\_struct() function are initialized with the values of the actual arguments.
- From lines 25-27, three printf() statement prints name, roll\_no and marks of the student.
- The most important thing to note about this program is that stu.name is passed as a

reference because name of the array is a constant pointer. So the formal argument of print\_struct() function i.e name and stu.name both are pointing to the same array. As a result, any changes made by the function print\_struct() will affect the original array. We can verify this fact by making the following amendments to our program.

### **Passing Structure variable as argument to a Function**

In the earlier section, we have learned how to pass structure members as arguments to a function. If a structure contains two-three members then we can easily pass them to function but what if there are 9-10 or more members? Certainly passing 9-10 members is a tiresome and error-prone process. So in such cases instead of passing members individually, we can pass structure variable itself.

The following program demonstrates how we can pass structure variable as an argument to the function.

```
1 #include<stdio.h>
2
3 /*
4 structure is defined above all functions so it is global.
5 */
6
7 structstudent
8 {
9 charname[20];
10 introll_no;
11 intmarks;
12 };
13
14 voidprint_struct(structstudentstu);
15
16 intmain()
17 {
18 structstudentstu={"George",10,69};
19 print_struct(stu);
20 return0;
21 }
22
23 voidprint_struct(structstudentstu)
24 {
25 printf("Name: %s\n",stu.name);
26 printf("Roll no: %d\n",stu.roll_no);
27 printf("Marks: %d\n",stu.marks);
```

```
28 printf("\n");
29 }
```

When the above code is compiled and executed, it produces the following result

Name: George

Roll no: 10

Marks: 69

### Flow of Execution of the above code

- In lines 7-12, a structure student is declared with three members namely: name, roll\_no and marks.
- In line 14, the prototype of function print\_struct() is declared which accepts an argument of type struct student.
- In line 18, a structure variable stu of type struct student is declared and initialized.
- In line 19, print\_struct() function is called along with argument stu. Unlike arrays, the name of structure variable is not a pointer, so when we pass a structure variable to a function, the formal argument of print\_struct() is assigned a copy of the original structure. Both structures reside in different memory locations and hence they are completely independent of each other. Any changes made by function print\_struct() doesn't affect the original structure variable in the main() function.
- The printf() statements from lines 25-27 prints the details of the student.

### Returning Structure from Function

Just as we can return fundamental types and arrays, we can also return a structure from a function. To return a structure from a function we must specify the appropriate return type in the function definition and declaration. Consider the following example:

```
struct player check_health(struct player p);
{
    ...
}
```

This function accepts an argument of type struct player and returns an argument of type struct player.

The following program demonstrates how we can return a structure from a function.

```
1 #include<stdio.h>
2
```

```

3 /*
4 structure is defined above all functions so it is global.
5 */
6
7 structplayer
8 {
9 charname[20];
10 floatheight;
11 floatweight;
12 floatfees;
13 };
14
15 voidprint_struct(structplayerp);
16 structplayerdeduct_fees(structplayerp);
17
18 intmain()
19 {
20 structplayerp={"Joe",5.9,59,5000};
21 print_struct(p);
22 p=deduct_fees(p);
23 print_struct(p);
24
25 return0;
26 }
27
28 structplayerdeduct_fees(structplayerp)
29 {
30 p.fees-=1000;
31 returnp;
32 }
33
34 voidprint_struct(conststructplayerp)
35 {
36 printf("Name: %s\n",p.name);
37 printf("Height: %.2f\n",p.height);
38 printf("Weight: %.2f\n",p.weight);
39 printf("Fees: %.2f\n",p.fees);
40
41 printf("\n");
42 }

```

When the above code is compiled and executed, it produces the following result

Name: Joe

Height: 5.90

Weight: 59.00

Fees: 5000.00

Name: Joe

Height: 5.90

Weight: 59.00

Fees: 4000.00

### **Flow of Execution of the above code**

- In lines 7-13, a structure of type player is declared with 4 members namely name, height, weight and fees.
- In line 15, the prototype of print\_struct() is declared which accepts an argument of type struct player and returns nothing.
- In line 16, the prototype of deduct\_fees() is declared which accepts an argument of type struct player and returns a structure of type struct player.
- In line 20, a structure variable p of type struct player is declared and initialized.
- In line 21, the print\_struct() function is passed an argument of type struct player. The function prints the details of the player and passes the control back to main() function.
- In line 22, deduct\_fees() function is called with an argument of type struct player. The function decrements the fees of the player by 1000 using the statement.  
p.fees -= 1000;
- and then returns the structure variable p to the called function i.e main(), where it is assigned back to the variable p.
- In line 23, the print\_struct() is called again with the same argument as before to check whether the details have been modified by deduct\_fees() or not.
- After printing the details of the function the control passes back to main() function and the program terminates.
- 

### **Pointers to Structures**

You can define pointers to structures in the same way as you define pointer to any other variable

```
struct Books *struct_pointer;
```

Now, you can store the address of a structure variable in the above defined pointer variable. To find the address of a structure variable, place the '&'; operator before the structure's name as follows –

```
struct_pointer = &Book1;
```

To access the members of a structure using a pointer to that structure, you must use the → operator as follows –

```
struct_pointer->title;
```

The following program demonstrates how to access structure members using pointers.

```
#include <stdio.h>
#include <string.h>
struct Books{
char title[50];
char author[50];
char subject[100];
int book_id;
};

/* function declaration */ voidprintBook(struct Books *book ); int main( ) {
structBooksBook1; /* Declare Book1 of type Book */ structBooksBook2; /* Declare
Book2 of type Book*/

/* book 1 specification */
strcpy( Book1.title, "C Programming"); strcpy( Book1.author, "Nuha Ali");
strcpy( Book1.subject, "C Programming Tutorial"); Book1.book_id = 6495407;

/* book 2 specification */
strcpy( Book2.title, "Telecom Billing");
strcpy( Book2.author, "Zara Ali");
strcpy( Book2.subject, "Telecom Billing Tutorial");
Book2.book_id = 6495700;

/* print Book1 info by passing address of Book1 */ printBook(&Book1);
/* print Book2 info by passing address of Book2 */ printBook(&Book2);
return 0;
}

voidprintBook( struct Books *book )
{
printf( "Book title : %s\n", book->title);
printf( "Book author : %s\n", book->author);
printf( "Book subject : %s\n", book->subject);
printf( "Book book_id : %d\n", book->book_id);
}
```

When the above code is compiled and executed, it produces the following result – Book title : C  
Programming  
Book author :Nuha Ali

Book subject : C Programming Tutorial Book book\_id : 6495407

Book title : Telecom Billing Book author : Zara Ali

Book subject : Telecom Billing Tutorial Book book\_id : 6495700

### Self Referential Structures

Self Referential structures are those structures that have one or more pointers which point to the same type of structure, as their member.

## Self Referential Structures

```
struct node {  
    int data1;  
    char data2;  
    struct node* link;  
};
```

In other words, structures pointing to the same type of structures are self-referential in nature.

Example:

```
structnode {  
    intdata1;  
    chardata2;  
    structnode* link;  
};  
  
intmain()  
{  
    structnode ob;  
    return0;  
}
```

In the above example ‘link’ is a pointer to a structure of type ‘node’. Hence, the structure ‘node’ is a self-referential structure with ‘link’ as the referencing pointer. An important point to consider is that the pointer should be initialized properly before accessing, as by default it contains garbage value.

The following Program to demonstrate Self referential structure.

```

#include <stdio.h>

structnode {
    intdata1;
    chardata2;
    structnode* link;
};

Int main()
{
    structnode ob1; // Node1

    // Initialization
    ob1.link = NULL;
    ob1.data1 = 10;
    ob1.data2 = 20;

    struct node ob2; // Node2

    // Initialization
    ob2.link = NULL;
    ob2.data1 = 30;
    ob2.data2 = 40;

    // Linking ob1 and ob2
    ob1.link = &ob2;

    // Accessing data members of ob2 using ob1
    printf("%d", ob1.link->data1);
    printf("\n%d", ob1.link->data2);
    return0;
}

```

When the above code is compiled and executed, it produces the following result

30  
40

## Unions

A **union** is a special data type available in C that allows storing different data types in the same memory location. You can define a union with many members, but only one member can contain a value at any given time. Unions provide an efficient way of using the same memory location for multiple-

purpose.

## Defining a Union

To define a union, you must use the **union** statement in the same way as you did while defining a structure. The union statement defines a new data type with more than one member for your program. The format of the union statement is as follows—

```
union [union tag] { member definition; member definition;  
...  
member definition;  
} [one or more union variables];
```

The **union tag** is optional and each member definition is a normal variable definition, such as int i; or float f; or any other valid variable definition. At the end of the union's definition, before the final semicolon, you can specify one or more union variables but it is optional. Here is the way you would define a union type named Data having three members i, f, and str—

```
union Data { int i;  
float f;  
char str[20]  
} data;
```

Now, a variable of **Data** type can store an integer, a floating-point number, or a string of characters. It means a single variable, i.e., same memory location, can be used to store multiple types of data. You can use any built-in or user defined data types inside a union based on your requirement.

The memory occupied by a union will be large enough to hold the largest member of the union. For example, in the above example, Data type will occupy 20 bytes of memory space because this is the maximum space which can be occupied by a character string.

The following example displays the total memory size occupied by the above union –

```
#include <stdio.h>  
#include <string.h>  
union Data {  
int i; float f;  
char str[20];  
};  
  
Int main()  
{  
union Data data;  
printf("Memory size occupied by data: %d\n", sizeof(data));  
return 0;
```

```
}
```

When the above code is compiled and executed, it produces the following result – Memory size occupied by data: 20

### Accessing Union Members

To access any member of a union, we use the **member access operator (.)**. The member access operator is coded as a period between the union variable name and the union member that we wish to access. You would use the keyword **union** to define variables of union type.

The following example shows how to use unions in a program –

```
#include <stdio.h>
#include <string.h>
union Data {
int i;
float f;
char str[20];
};
int main()
{
union Data data;
data.i = 10;
data.f = 220.5;
strcpy(data.str, "C Programming");
printf("data.i : %d\n", data.i);
printf("data.f : %f\n", data.f);
printf("data.str : %s\n", data.str);
return 0;
}
```

When the above code is compiled and executed, it produces the following result – data.i : 1917853763

data.f : 4122360580327794860452759994368.000000

data.str : C Programming

Here, we can see that the values of **i** and **f** members of union got corrupted because the final value assigned to the variable has occupied the memory location and this is the reason that the value of **str** member is getting printed very well.

Now let's look into the same example once again where we will use one variable at a time which is the main purpose of having unions –

```
#include <stdio.h>
#include <string.h>
union Data
{
```

```

int i;
float f;
char str[20];
};
int main( )
{
union Data data;
data.i = 10;
printf( "data.i : %d\n", data.i);
data.f = 220.5;
printf( "data.f : %f\n", data.f);
strcpy(data.str, "C Programming");
printf( "data.str : %s\n", data.str);
return 0;
}

```

When the above code is compiled and executed, it produces the following result –

data.i : 10

data.str : C Programming

Here, all the members are getting printed very well because one member is being used at a time.

### **Enumerated data types**

Enumeration is a user defined data type in C language. It is used to assign names to the integral constants which makes a program easy to read and maintain. The keyword “enum” is used to declare an enumeration.

Here is the syntax of enum in C language,

```
Enum enum_name{const1, const2, ..... };
```

The enum keyword is also used to define the variables of enum type. There are two ways to define the variables of enum type as follows.

```
enum week{sunday, monday, tuesday, wednesday, thursday, friday, saturday};
enum week day;
```

The following example demonstrate the use of enum in the program

```
#include<stdio.h>
enum week{Mon=10, Tue, Wed, Thur, Fri=10, Sat=16, Sun};
enum day{Mond, Tues, Wedn, Thurs, Frid=18, Satu=11, Sund};
int main()
{
```

```

printf("The value of enum week: %d\t%d\t%d\t%d\t%d\t%d\t%d\n\n",Mon , Tue, Wed, Thur,
Fri, Sat, Sun);
printf("The default value of enum day: %d\t%d\t%d\t%d\t%d\t%d\t%d",Mond , Tues, Wedn,
Thurs, Frid, Satu, Sund);
return 0;
}

```

When the above code is compiled and executed, it produces the following result –

The value of enum week: 10111213101617

The default value of enum day: 0123181112

In the above program, two enums are declared as week and day outside the main() function. In the main() function, the values of enum elements are printed.

#### **Summary exercise:**

1. Structures Declaration and Usage
2. Usage of Structures for Complex Problems
3. Unions and their Importance, difference from structures

#### **Review and revision:**

##### **Review and revision exercise (1) :**

Look over your lecture notes and create a quiz and/or discussion exercise based on the content. Doing this will help when students are preparing for exams.

#### **Lecture topic quiz / MCQs:**

1. Which of the following is true for definition of a structure \_\_\_\_\_  
 A) Items of the same data type                      B) **Items of the different data type**  
 C)Integers with user defined names              D) List of Strings  
 L1
2. The keyword used to define a structure is \_\_\_\_\_  
 A) stru    B)**struct**    C)structure                                      D)STRUC  
 L1
3. The operator used to access the structure member is \_\_\_\_\_  
 A)\*    B)&    C).    D)|  
 L1
4. The operator exclusively used with pointer to structure is \_\_\_\_\_  
 A.).    B)[]    C)**→**    D)\*  
 L

5. Which of the following is correct for a Structure definition?
- A) Scalar data type
  - B) C) Enumerated type
  - B) Derived data type**
  - D) Null Type
- L1
6. When accessing a structure member, the identifier to the left of the dot operator is
- A) A structure ember
  - B) The structure tag
  - C) A structure variable**
  - D) The keyword struct
- L2
7. When a structure is an element to another structure, it is called asa \_\_\_\_\_
- A) Union**
  - B) Structure within a structure**
  - C) Pointer to Structure
  - D) Array of Structures
- L1
8. A \_\_\_\_\_ structure is one which contains a pointer to its own type.
- A) Self-referential
  - B) Nested
  - B) C) Array
  - D) Pointer
- L1
- C)
9. Consider the following declaration of union st
- ```
{
    char c;
    int x;
    float y;
}p;
```
- How many bytes are allocated to union variable p?
- A) 7bytes
  - B)4bytes**
  - C)1byte
  - D) 2bytes
- L2
10. The size of structure and union is same when they contain \_\_\_\_\_
- A) Single member**
  - B) any number of members
  - C) Arrays of different types
  - D) Pointers to different types
- L2
11. The operator used to find the size of any variable\_\_\_\_\_
- A) sizeof()**
  - B)sizof()
  - C) sizeof()
  - D) size()
- L1
12. The operator → is same as the combination of the operators \_\_\_\_\_
- A) \*and .
  - B) &and.
  - C) \*and &
  - D) & and |
- L1
13. Union can store \_\_\_\_\_ number of values at a time

- A) All its members    B) Only 1  
 B) C) 2                      D) Cannot hold value  
 L1
14. 'C' provides a facility for user defined data type using \_\_\_\_\_ concept  
 A) Array                      B) Function                      C) Pointer                      D) Structure L1
15. In the expression  $p \rightarrow$  value, p is a  
 A) Address                      B) Pointer                      C) Structure                      D) Header  
 L1
16. In C language the expression  $(*ps).x$  is equal to \_\_\_\_\_  
 A)  $ps \rightarrow x$                       B)  $x \rightarrow ps$                       C)  $ps \rightarrow *x$                       D) None  
 L1
17. Which of the following is a list of named integer constants?  
 A) typedef                      B) enumeration                      C) structure                      D) union  
 L1
18. Which of the following is a memory location that is shared by two or more different types of variables?  
 A) typedef                      B) enumeration                      C) structure                      D) union  
 L2

#### **Review and revision exercise (2) :**

#### **Discussion questions / Expected Questions:**

#### **Short Questions :**

1. Define Structure? How to Initialize a Structure?                      L2  
 2. How to represent self-referential structures?                      L2  
 3. Define Union? How to represent an union?                      L2  
 4. Write some of the differences between Structure and Union?                      L1  
 5. What are the Different ways of representing Structures and Functions?                      L2

#### **Long Questions :**

1. Define Structure and write the general syntax for declaring and accessing members. L1  
 2. How to copy and compare structure variables? Illustrate with example.                      L2  
 3. Write a C program that defines a structure employee containing the details such as empno,emp name, department name and salary. The structure has to store 20 employees in an organization. Use the appropriate method to define the above details and define a function that will display the contents?                      L3  
 4. Explain the Nested structures  
 L2  
 5. Write a C program to read and display student details using structure.                      L3  
 6. Define union. Give the general template for union.                      L1  
 7. List out the differences between unions, structures and arrays  
 L4

- |                                                                               |    |
|-------------------------------------------------------------------------------|----|
| 8. How data elements are stored under unions, explain with example?           | L3 |
| 9. Write a C program to illustrate the concept of structure within structure. | L3 |

- 8. How data elements are stored under unions, explain with example?
- 9. Write a C program to illustrate the concept of structure within structure.

**Course Code: 20CS C01**

**Course Title: PROGRAMMING FOR PROBLEM SOLVING**

**UNIT-V**

**Objective:**

The primary goals of the module/unit are to introduce:

- File concepts and how to read from a file which is stored in our computer and how to write data to a file in a computer.

**Outcome:**

On Successful completion of the course, students will be able to develop applications with files.

**Scope:**

The Module/Unit covers the following topics:

Files: Introduction to files, file operations, reading data from files, writing data to files, error handling during file operations – Basic files concepts and manipulation of data stored in a file.

While the topics are taught using a C language, the intent of the course is to teach a programming methodology to handle files and also a laboratory component that involves development and testing of programs to manipulate data in files.

**Text and Reference:**

**(a) Text Book:**

**TB1.** M.T. Somashekhar “Problem Solving with C”, 2nd Edition, Prentice Hall India Learning Private Limited 2018

**TB2.** A K Sharma “Computer Fundamentals and Programming”, 2<sup>nd</sup> Edition, University Press, 2018

**TB3.** Pradeep Dey and Manas Ghosh, “Programming in C”, Oxford Press, 2<sup>nd</sup> Edition, 2017

**(b) Reference Books:**

**RB1.** Byron Gottfried , Schaum’s “Outline of Programming with C”, McGraw- Hill

**RB2.**Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Prentice Hall of India.

**RB3.** E. Balaguruswamy, Programming in ANSI C, Tata McGraw-Hill.

**RB4.** ReemaTharaja "Introduction to C Programming", Second Edition, OXFORD Press, 2015

**RB5:** <https://www.tutorialspoint.com/cprogramming/index.htm>

| <b>Module /Lecture</b> | <b>Theme</b>                                                  | <b>Learning Objectives</b>                                                          |
|------------------------|---------------------------------------------------------------|-------------------------------------------------------------------------------------|
| I                      | Introduction to files, file operations                        | To understand the basics of files and all possible operations on files              |
| II                     | Reading data from files                                       | To understand how to read data from a file.                                         |
| III                    | Writing data to files, error handling during file operations. | To understand how to write data to a file with different error handling approaches. |

|                      |      |
|----------------------|------|
| <b>Lecture title</b> | L.37 |
| <b>Date</b>          |      |

**Key topics / themes of the lecture :** Introduction to files, file operations

#### **INTRODUCTION TO FILES:**

In real world application when data is large then it is necessary to store it on the disk in files. A file represents a sequence of bytes. A file is created in secondary storage. Files are of two types text file and binary file.

C language provides access to files with high/low level functions. FILE is a structure defined in standard input/output library.

The file pointer is created and used as **FILE \*fp;** where fp is a file pointer

#### **Types of files**

##### **1. Text File :**

A text stream consists of a sequence of characters. They are grouped in lines. Stores the data in the text form

##### **2. Binary File :**

It stores the data in the binary form (1's and 0's). A binary stream consists of a sequence of data. They are stored in their memory representation.

#### **Operations on Files**

C language supports a number of functions to perform basic operations on files

- creating a file
- opening a file

- writing into the file
- reading from the file
- close the file

#### **files I/O functions to manipulate data in file :**

**fopen()** : create a file or open an existing file  
**fclose()** : close a file  
**getc()** : read a character  
**putc()** : write a character  
**fprintf()** : write a formatted values  
**fscanf()** : read set of values  
**getw()** : read an integer  
**putw()** : write an integer

#### **How to open a file?**

**fopen()** function is used to open a file . This function creates a new file if the file does not exist. .

#### **Syntax: fopen(filename,mode)**

Here, the first argument is file name and second argument is access mode.

The following are the different access modes in C,

r : read mode  
 w : writing mode  
 a : append mode  
 r+ : First reading then writing  
 w+: reading and writing and content is truncated  
 a+ : reading and writing and content is not truncated

#### **Example: Demonstrate the opening of a file**

```
#include<stdio.h>
int main()
{
    FILE *fp; //creating a file pointer
    fp=fopen("abc.txt","r"); // open a file
    return 0;
}
```

#### **How to close a file?**

**fclose()**is used for closing the file. The **fclose()** function returns zero on success, or EOF if there is an error in closing the file.

#### **Syntax: fclose(filepointer)**

#### **Example:**

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

```

FILE *fp; //creating a file pointer
fp=fopen("abc.txt","r"); // open a file
fclose(fp); // at the end close the file
return 0;
}

```

**Summary exercise:**

1. File pointer is created using FILE structure which is defined in stdio.h header file.
2. There 2 types of files: Text file, binary file
3. files can be opened in different modes to manipulate data using fopen() function.

**Review and revision:**

**Review and revision exercise (1) :**

Look over your lecture notes and create a quiz and/or discussion exercise based on the content. Doing this will help when students are preparing for exams.

**Lecture topic quiz / MCQs:**

1. Choose correct syntax for opening a file. (CO6)(BT1)
  - a) FILE \*fopen(const \*filename, const char \*mode)
  - b) FILE \*fopen(const \*filename)
  - c) FILE \*open(const \*filename, const char \*mode)
  - d) FILE open(const\*filename)

Answer: a) FILE \*fopen(const \*filename, const char \*mode)
2. Outline the function of the mode ' w+'? (CO6) (BT 2)
  - a) create text file for writing, discard previous contents if any
  - b) create text file for update, discard previous contents if any
  - c) create text file for writing, do not discard previous contents if any
  - d) create text file for update, do not discard previous contents if any

Answer:b) create text file for update, discard previous contents if any
3. Before we can read or write to a file in C, what do we need to do? (CO6)(BT 1)
  - a) open the file
  - b) close the file
  - c) print the content of the file
  - d) NONE

Answer:a) open the file
4. Which flag is used in fopen function to append data to an existing file instead of recreating the file?(CO6)(BT 1)
  - a) a
  - b) r
  - c) w
  - d) W+

Answer:a) a

5. If the mode includes b after the initial letter in fopen function then what does it indicates?(CO6)(BT 1)

- a) text file
- b) big text file
- c) binary file
- d) empty text

Answer: c) binary file

6. Select the correct statement about FILE \*fp. (CO6)(BT 3)

- a) FILE is a keyword in C for representing files and fp is a variable of FILE type.
- b) FILE is a stream
- c) FILE is a buffered stream
- d) FILE is a structure and fp is a pointer to the structure of FILE type

Answer: d) FILE is a structure and fp is a pointer to the structure of FILE type

7. Summarize the first and second arguments of fopen().(CO6)(BT 2)

- a) name of the file,mode
- b) name of the user,mode
- c) file pointer , mode
- d) None

Answer:a) a) name of the file,mode

8. Find the type of FILE. (CO6) (BT1)

- a) int type
- b) char \* type
- c) struct type
- d) float

Answer:c) struct type

9. For binary files, which of the following must be appended to the mode string. (CO6)(BT 1)

- a) b
- b) B
- c) binary
- d) 01

Answer: a) b

10. Select the reason for closing a file in C language.(CO6)(BT3)

- a) fclose(fp) closes a file to release the memory used in opening a file.
- b) Closing a file clears Buffer contents from RAM or memory.
- c) Unclosed files occupy memory and PC hangs when on low memory.

d) All of these

Answer: d) All of these

### Review and revision exercise (2) :

#### Discussion questions / Expected Questions:

1. Compare and contrast Text file and binary file.(CO6)(BT3)
2. Explain different modes to open a text file. (CO6)(BT2)
3. How does file pointer created in C program? (CO6)(BT1)
4. Why do we need a file? (CO6)(BT1)

|               |      |
|---------------|------|
| Lecture title | L.38 |
| Date          |      |

**Key topics / themes of the lecture :**     Reading data from files

We can read both text files as well as binary files.

#### Reading data from text file:

The following 3 functions are used for reading data from a Text File.

##### 1. **fgetc():**

The fgetc() function reads a character from the input file referenced by file pointer. The return value is the character read, or in case of any error, it returns EOF.

**Syntax:** **int fgetc( FILE \* fp );**

**Example:** Write a program to read data from file using fgetc() method

```
#include <stdio.h>
int main () {
    FILE *fp;
    int c;
    fp = fopen("myfile.txt","r");
    do {
        c = fgetc(fp);
        if( c==EOF ) {
            break ;
        }
        printf("%c", c);
    } while(1);
```

```
        fclose(fp);
        return(0);
    }
```

## 2. fgets():

It reads a string from a stream.

**Syntax:** `char *fgets( char *buf, int n, FILE *fp );`

The functions fgets() reads up to n-1 characters from the input stream referenced by file pointer fp. It copies the read string into the buffer buf, appending a null character to terminate the string.

If this function encounters a newline character '\n' or the end of the file EOF before they have read the maximum number of characters, then it returns only the characters read up to that point including the new line character.

**Example:** Write a program to read data from file using fgets() method

```
#include <stdio.h>
void main() {
    FILE *fp;
    char output[255];
    fp = fopen("myfile.txt", "r");
    fgets(buff, 255, fp);
    printf("%s\n", output );
    fclose(fp);

}
```

## 3. fscanf()

This function is to read strings from a file, but it stops reading after encountering the first space character.

**Syntax:** `int fscanf(FILE *fp, const char *format, [ argument, ...])`

**Example:** Write a program to read data from file using fscanf() method

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

    FILE *fp;
    char name[255];
    int rollnum;
    fp = fopen("myfile.txt", "r");
    fscanf(fp, "%s", name);
    fscanf(fp, "%d", rollnum);

    printf(" Name: %s\nRoll No.:%d", name, rollnum );
    fclose(fp);
```

}

### **Reading data from binary file:**

The following function is used for reading data from a Binary File.

#### **1. `fread()`**

Reads data from the given stream into the array .

**Syntax:** `size_t fread(void *ptr, size_t sizeOfElements, size_t numberOfElements, FILE *fp);`

Here,

`ptr` – the pointer to a block of memory with a minimum size of `sizeOfElements * numberOfElements` bytes.

`sizeOfElements` – the size in bytes of each element to be read.

`numberOfElements` – the number of elements, each one with a size of `size` bytes.

`fp` – the pointer to a `FILE` object that specifies an input stream.

**Example:** Write a program to read data from file using `fread()` method

```
#include <stdio.h>
int main () {
    FILE *fp;
    char output[100];
    /* Open file for reading */
    fp = fopen("myfile.txt", "r");

    /* Read and display data */
    fread(output, 1, 100, fp);
    printf("%s\n", output);
    fclose(fp);
    return 0;
}
```

### **Summary exercise:**

[At the end of the lecture, briefly look back over your notes and give summarizing the top three ‘take-away’ points (these might include key knowledge and/or any suggested further reading the lecturer recommends etc)]:

1. single character, a word or a line can be read at a time from a file.
2. `fgetc()`, `fgets()` and `fscanf()` methods are used to read data from text file
3. `fread()` method is used for reading data from binary file

### **Review and revision:**

#### **Review and revision exercise (1) :**

Look over your lecture notes and create a quiz and/or discussion exercise based on the content.

Doing this will help when students are preparing for exams.

**Lecture topic quiz / MCQs:**

1. choose the correct difference between fscanf() and scanf() methods (CO6) (BT 3)
  - a) fscanf() can read from standard input whereas scanf() specifies a stream from which to read
  - b) fscanf() can specifies a stream from which to read whereas scanf() can read only from standard input
  - c) fscanf() and scanf() has no difference in their functions
  - d) fscanf() and scanf() can read from specified stream

Answer:b) fscanf() can specifies a stream from which to read whereas scanf() can read only from standard input
2. What is the value of EOF which is defined in stdio.h header file?(CO6) (BT 1)
  - a) 1
  - b) 0
  - c) NULL
  - d) -1

Answer: d) -1
3. Infer the statement: "fwrite() can be used only with files that are opened in binary mode".(CO6) (BT 2)
  - a) true
  - b) false

Answer: a) true
4. Recall the function of fputs()?(CO6)(BT1)
  - a) read a line from a file
  - b) read a character from a file
  - c) write a character to a file
  - d) write a line to a file

Answer:d) write a line to a file
5. Summarize the action done by the following C code snippet.(CO6)(BT1)

```
char *gets(char *s)
```

  - a) reads the next input line into the array s
  - b) writes the line into the array s
  - c) reads the next input character into the array s
  - d) write a character into the array

Answer:a) reads the next input line into the array s

**Review and revision exercise (2) :**

**Discussion questions / Expected Questions:**

1. Outline the functions which are used to read data from a text file. (CO6)(BT2)
2. Infer the statement: "Reading from a text file and reading from a binary file can be done using same function X". Are you agreeing with this statement? Summarize your opinion. Outline the functions which are used to read data from a text file. (CO6)(BT2)
3. Develop a C program to read from text file which contains student name, roll no and marks of 3 subjects. Calculate and Display the percentage of marks for each student. (CO6)(BT3)

|                      |      |
|----------------------|------|
| <b>Lecture title</b> | L.39 |
| <b>Date</b>          |      |

**Key topics / themes of the lecture :** Writing data to files, error handing during file operations

**WRITING DATA TO FILES**

We can write data to both text files as well as binary files.

**Writing data to a text file:**

The following 3 functions are used for writing data to a Text File.

**1. fputc():**

The fputc() function writes the character value of the argument c to the output stream referenced by file pointer. It returns the written character written on success otherwise EOF if there is an error.

**Syntax:** **int fputc(int c, FILE \* fp );**

**Example:** Write a program to write data to a file using fputc() method

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

```
int main () {
    FILE *fp;
    int ch;

    fp = fopen("myfile.txt", "w");
    for( ch = 65 ; ch <= 92; ch++ ) {
        fputc(ch, fp);
    }
    fclose(fp);
```

```
        return 0 ;
    }
```

## 2. fputs():

It writes the string to the output stream referenced by file pointer.

**Syntax:** `char *fputs( char *str, FILE *fp );`

**Example:** Write a program to write data to a file using fputs() method

```
#include <stdio.h>

main() {
    FILE *fp;

    fp = fopen("myfile.txt", "w");
    fputs("This is testing for fputs...\\n", fp);
    fclose(fp);
}
```

## 3. fprintf()

It sends formatted output to a stream..

**Syntax:** `int fprintf(FILE *fp, const char *format, [ argument, ...])`

**Example:** Demonstrate a program to write data to a file using fprintf() method

```
#include <stdio.h>
#include <stdlib.h>
int main () {
    FILE * fp;
    fp = fopen ("file.txt", "w");
    fprintf(fp, "%s %d", "CSE dept code is", 733);
    fclose(fp);
    return 0;
}
```

## writing data to binary file:

The following function is used for writing data to a Binary File.

### 1. fwrite()

It writes data from the array pointed to by ptr to the given stream.

**Syntax:** `size_t fread(void *ptr, size_t sizeOfElements, size_t numberOfElements, FILE *fp);`

Here,

`ptr` – the pointer to a block of memory with a minimum size of `sizeOfElements` \*

numberOfElements bytes.  
sizeOfElements – the size in bytes of each element to be read.  
numberOfElements – the number of elements, each one with a size of size bytes.  
fp – the pointer to a FILE object that specifies an input stream.

**Example:** Demonstrate a program to write data to a file using fwrite() method

```
#include<stdio.h>
int main () {
    FILE *fp;
    char str[] = "CSE department code is 733";
    fp = fopen( "file.txt" , "w" );
    fwrite(str , 1 , sizeof(str) , fp );
    fclose(fp);
    return 0;
}
```

### **RANDOM FILE ACCESS**

We can take the file pointer to any part of the file for reading or writing. This can be done using the following functions:

**1. fseek():**

It sets the file position of the stream to the given offset.

**Syntax:** **int fseek(FILE \*fp, long int offset, int whence)**

Where, fp - the pointer to a FILE object that identifies the stream.

offset – number of bytes to offset from whence.

whence – the position from where offset is added.

- It is specified by one of the following constants:

SEEK\_SET - represents Beginning of file

SEEK\_CUR - represents Current position of the file pointer

SEEK\_END - represents End of file

**2. tell():**

It returns the current file position of the given stream.

**Syntax:** **long int tell(FILE \*filepointer)**

**3. rewind():**

It sets the file position to the beginning of the file of the given stream.

**Syntax:** **void rewind(FILE \*stream)**

**Example:** Demonstrate a program to access data from a file using random access file operations.

```
#include <stdio.h>
int main () {
    FILE *fp;
    int len;
    fp = fopen("file.txt", "r");
```

```

fseek(fp, 0, SEEK_END);
len = ftell(fp);
fclose(fp);
printf("Total size of file.txt = %d bytes\n", len);
return 0;
}

```

## **ERROR HANDLING DURING FILE OPERATIONS**

The following file status functions are used for handling errors during file operation:

1. **feof()**: it is used to check if end of the file has been reached  
**syntax:** **int feof(FILE \*fp);**
2. **ferror()**: it is used to check the error status of the file. it returns true if it finds error, otherwise false.  
**syntax:** **int ferror(FILE \*fp);**
3. **clearerr()**: this function is used to change the status of the file from error state  
**syntax:** **void clearerr(FILE \*fp);**

### **Summary exercise:**

[At the end of the lecture, briefly look back over your notes and give summarizing the top three ‘take-away’ points (these might include key knowledge and/or any suggested further reading the lecturer recommends etc)]:

1. fputc(), fprintf() and fwrite() methods are used for writing data to a file
2. fseek(),ftell() and rewind() methods used to access data from a file at random locations
3. feof() function is used for checking the end of file status

### **Review and revision:**

#### **Review and revision exercise (1) :**

Look over your lecture notes and create a quiz and/or discussion exercise based on the content. Doing this will help when students are preparing for exams.

#### **Lecture topic quiz / MCQs:**

1. Tell the function which will return the current file position for stream?(CO6)(BT1)
  - a) fgetpos()
  - b) fseek()
  - c) ftell()
  - d) fsetpos()

Answer:c ftell()

2. Identify the main reason to prefer fseek() over rewind().(CO6) (BT3)
  - a) rewind() doesn't work for empty files
  - b) rewind() may fail for large files
  - c) In rewind, there is no way to check if the operations completed successfully
  - d) All of the above

Answer:c In rewind, there is no way to check if the operations completed successfully

3. Choose the correct C functions to read or write to a Text file. (CO6)(BT1)

- a) fprintf(), fscanf()
- b) fread(), fwrite()
- c) fprint(), scanf()
- d) read(), write()

Answer:a) fprintf(), fscanf()

4. Choose the correct C functions to read or write to a binary file. (CO6)(BT1)

- a) fprintf(), fscanf()
- b) fread(), fwrite()
- c) fprint(), scanf()
- d) read(), write()

Answer:b) fread(), fwrite()

5. Which C function is used to move current pointer to the beginning of file.?(CO6)(BT1)

- a) revise(fp)
- b) rewind(fp)
- c) ftell(fp)
- d) frewind(fp)

Answer:b) rewind(fp)

#### **Review and revision exercise (2) :**

#### **Discussion questions / Expected Questions:**

1. Discuss in detail how files are accessed randomly in C with suitable example. (CO6)(BT3)
2. Develop a C program to copy the content of one file to another file. (CO6)(BT3)
3. Show how to write data to a binary file. (CO6)(BT2)



# CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. [www.cbit.ac.in](http://www.cbit.ac.in)

Affiliated to  
University of Hyderabad



Accredited by  
NAAC  
Grade - A++ Rank 4



100th Rank In  
NATIONAL INSTITUTE RANKING  
2018 - 2019



COMMITTED TO  
RESEARCH,  
INNOVATION AND  
EDUCATION

41  
years

**Course Code:** 20CSC01

**Course Title:** PROGRAMMING FOR PROBLEM SOLVING

**The primary goals of the module/unit are to introduce**

- Basic representation of preprocessor directives and its types.
- To includes files directive, error directive and macro directive .

**Outcome:**

1. Develop applications using file I/O(inclusion of file directive)

**Scope:**

The Module/Unit covers the following topics:

Types of preprocessor directives, examples.

**Text and Reference:**

**(a) Text Book:**

1. M.T. Somashekhar "Problem Solving with C", 2<sup>nd</sup> Edition, Prentice Hall India Learning Private Limited 2018
2. AKSharma "ComputerFundamentalsandProgramming", 2<sup>nd</sup> Edition, University Press, 2018
3. PradeepDeyandManasGhosh, "ProgramminginC", OxfordPress, 2<sup>nd</sup> Edition, 2017

**(b) Reference Books:**

1. Byron Gottfried, Schaum's "Outline of Programming with C", McGraw- Hill.
2. Brian W. Kernighan and Dennis M. Ritchie, The C Programming Language, Prentice Hall of India.
3. E. Balaguruswamy, Programming in ANSI C, Tata McGraw-Hill.
4. Reema Tharaja "Introduction to C Programming", Second Edition, OXFORD Press, 2015.

| Module/<br>Lecture | Theme                                       | Learning Objectives                                                  |
|--------------------|---------------------------------------------|----------------------------------------------------------------------|
| 40                 | Types of preprocessor directives, examples. | Able to learn preprocessor directives and structure of preprocessing |

|               |                                                                          |
|---------------|--------------------------------------------------------------------------|
| Lecture title | L40-Preprocessor Directives: Types of preprocessor directives, examples. |
| Date          |                                                                          |

## Lecture Notes

### PRE PROCESSOR DIRECTIVES

The C preprocessor is a macro processor that is used automatically by the C compiler to transform programmer defined programs before actual compilation takes place. It is called a macro processor because it allows the user to define macros, which are short abbreviations for longer constructs. So, it can be also be referred to as pre compiled fragments of code. Some possible actions are the inclusions of other files in the file being compiled, definitions of symbolic constants and macros and

conditional compilation of program code and conditional execution of preprocessor directives.

List of Preprocessor directives which starts with #: **#include, #define, #undef, #ifdef, #ifndef, #if, #else, #elif, #endif, #error, #pragma**

#### **# include** preprocessor directive:

Only defined at the top of the program definition and only white space and comments may appear before preprocessor directive line. This directive includes a copy of the specified file or library. And is written like so-

**# include <filename>**  
or

**#include "file name",**

If included inside <> then the compiler looks for the file in an implementation defined manner (or system directory paths). If included inside a quote “ ” statement then it searches for the file in the same directory as the program being written.

So if we include a file named file1 while writing a program name code.c which is being stored in a folder called test. Then inclusion like so -- #include “file1” will tell the compiler to look for the file inside the test folder, It looks at neighboring folders or subfolders too but it starts its search in the test folder.(This sub search option is compiler dependent).

A macro is a segment of code which is replaced by the value of macro. Macro is defined by #define directive. There are two types of macros: Object-like Macros, Function-like Macros

### **Object-like Macros (Symbolic constants)**

The object-like macro is an identifier that is replaced by value. It is widely used to represent numeric constants. Used to create symbolic constants and macros, meaning useful for renaming long named data types and certain constants or values which is being used throughout the program definition.

**#define identifier replacement-text**

When this line appears in a file, all subsequent occurrences of identifier that do not appear in string literals will be replaced by the replacement text automatically before program compilation takes place.

**For example: #define PI 3.14159**

Replaces all subsequent occurrences of the symbolic constant PI with numeric constant 3.14159. Symbolic constants enable the programmer to create a name for a constant and use that name throughout program execution. If this needs to be modified ,it has to be done in the #define directive statement. After recompilation all definitions get modified accordingly.

### **Function-like Macros**

The function-like macro looks like function call. For example:

**#define MIN(a,b) ((a)<(b)?(a):(b))**

Here, MIN is the macro name.

Example for Function-like Macros :

```
#include <stdio.h>
#define MIN(a,b) ((a)<(b)?(a):(b))
void main() {
    printf("Minimum between 10 and 20 is: %d\n", MIN(10,20));
}
```

**Output:**

Minimum between 10 and 20 is: 10

**Differences between MACRO and FUNCTION:**

| Macro                                                    | Function                                               |
|----------------------------------------------------------|--------------------------------------------------------|
| Macro is <b>Preprocessed</b>                             | Function is <b>Compiled</b>                            |
| <b>No Type Checking</b>                                  | <b>Type Checking</b> is Done                           |
| <b>Code Length Increases</b>                             | <b>Code Length remains Same</b>                        |
| Speed of Execution is <b>Faster</b>                      | Speed of Execution is <b>Slower</b>                    |
| Before Compilation macro name is replaced by macro value | During function call , Transfer of Control takes place |
| Useful where small code appears many time                | Useful where large code appears many time              |
| Generally Macros do not extend beyond one line           | Function can be of any number of lines                 |
| Macro does not Check <b>Compile Errors</b>               | Function Checks <b>Compile Errors</b>                  |

### Preprocessor Formatting

A preprocessing directive cannot be more than one line in normal circumstances. It may be split cosmetically with Backslash-Newline. Comments containing Newlines can also divide the directive into multiple lines.

for example, you can split a line cosmetically with Backslash-Newline anywhere:

```
/*
 */ # /*
 */
defi\
ne FO\
O 10\
20
is equivalent into #define FOO 1020.
```

**#undef**

To undefine a macro means to cancel its definition. This is done with the **#undef** directive.

**Syntax:**

**#undef token**

**define and undefine example**

```
#include <stdio.h>
#define PI 3.1415
#undef PI
```

```
main() {
    printf("%f",PI);
}
```

#### **Output:**

Compile Time Error: 'PI' undeclared

#### **#ifdef**

The #ifdef preprocessor directive checks if macro is defined by #define. If yes, it executes the code.

#### **Syntax:**

```
#ifdef MACRO
//cod
e
#endif
```

#### **#ifndef**

The #ifndef preprocessor directive checks if macro is not defined by #define. If yes, it executes the code.

#### **Syntax:**

```
#ifndef MACRO
//cod
e
#endif
```

#### **#if**

The #if preprocessor directive evaluates the expression or condition. If condition is true, it executes the code.

#### **Syntax:**

```
#if expression
//cod
e
#endif
```

#### **#else**

The #else preprocessor directive evaluates the expression or condition if condition of #if is false. It can be used with #if, #elif, #ifdef and #ifndef directives.

#### **Syntax:**

#### **Example**

#### **Syntax with #elif**

```
#if expression      e #else
'                  //else
code
#endif

#include <stdio.h>
#include <conio.h>
#define NUMBER 1
void main() {
#if NUMBER==0
printf("Value of Number is: %d",NUMBER);
```

```
#else
print("Value of Number is non-zero");
#endif
getch();
}
```

#### Output

Value of Number is non-zero

#### Example Code:

```
#include<stdio.h> -- Include external source code file
#include "file1" - Include a file which is in the same folder as the parent file
#define BUFFER_SIZE 250
int main(){

    char array[BUFFER_SIZE] = {0}; int i
    = 9;

    printf("%d",i);

    printf("%s",array);
    return 0;

}//end main body
```

#### Conditional Compilation:

Conditional compilation enables the coder to control the execution of preprocessor directives and the compilation of program code. Conditional preprocessor directives evaluate constant integer expressions. The ones that cannot be evaluated in preprocessor directives are sizeof expressions and enumeration constants.

Every #if construct ends with #endif

*Example –*

```
#if !defined(MY_CONSTANT)
#define MY_CONSTANT 0
#endif
```

These directives determine if *MY\_CONSTANT* is *defined* or not. The expression defined (*MY\_CONSTANT*) evaluates to 1 if *MY\_CONSTANT* is not defined and is defined else it evaluates to 0.

**Example:** conditional compilation

```
#include <stdio.h>

#define MEMBER

int main()

{

float amount,total;

printf("enter amount\n");

scanf("%f",&amount);

#ifndef MEMBER

total= amount*0.8;

printf("Amount to be paid=%f",total);

#else

printf("Amount to be paid=%f",amount);

#endif

}
```

**Summary exercise:**

Following table will show you various directives that we have studied in this chapter:

| Directives | Description                                                                |
|------------|----------------------------------------------------------------------------|
| #define    | It substitutes a preprocessor macro.                                       |
| #include   | It inserts a particular header file from another file.                     |
| #undef     | A preprocessor macro is undefined.                                         |
| #ifdef     | It returns true if the macro is defined.                                   |
| #ifndef    | It returns true if the macro is not defined.                               |
| #if        | It tests if the compile time condition is true.                            |
| #else      | It is an alternative for #if.                                              |
| #elif      | It has #else and #if in one statement.                                     |
| #endif     | The conditional preprocessor is ended.                                     |
| #error     | It prints the error message on stderr.                                     |
| #pragma    | It issues special commands to the compiler by using a standardized method. |

**Questions:**

1. What are preprocessor commands? Give examples.
2. Difference between Macro and Function.
3. Distinguish between function and preprocessor directive.
4. Difference between #ifdef and #ifndef with example .
5. Give a brief note about Error directive.

## MCQ

1. What will be the output of the program?

```
#include<stdio.h>

#define int char

void main()
{
    int i = 65;

    printf("sizeof(i)=%d", sizeof(i));
}
```

- A. Sizeof(i)=2;
- B. Sizeof(i)=1;
- C. Compilation error
- D. None

Answer:B

2. What will be the output of the program?

```
#include<stdio.h>
#define a 10
void main()
{
#define a 50
printf("%d", a);
}
```

- A. 50
- B. 10
- C. Compilation error
- D. None

Answer: A

3. What will be the output of the program?

```
#include <stdio.h>
#define p 17;
int main()
{
    printf("%d", p);
    return 0;
}
```

- A. 17
- B. Run-time error
- C. Compilation error

D. Garbage value

Answer: A

4. In which stage the following code

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

gets replaced by the contents of the file *stdio.h*

- A. During editing
- B. During linking
- C. During Preprocessor
- D. During execution

Answer: C

5. What will be the output of the program?

```
#include<stdio.h>
#define SQR(x) (x*x)
int main()
{
    int a, b=3;
    a = SQR(b+2);
    printf("%d\n", a);
    return 0;
}
```

- A. 25
- B. 11
- C. Error
- D. Garbage value

Answer: B



**Course Code: 20CS C01**

**Course Title: PROGRAMMING FOR PROBLEM SOLVING**

## **UNIT-I**

1. Draw and explain the block diagram of computer. [R][CO1]
2. Define algorithm and its characteristics. Write algorithm for finding factorial of a number.[U][CO1]
3. What is flowchart? Explain different symbols used for flowchart.[R][CO2]
4. Draw the flowchart to find the greatest of three numbers.[U][CO2]
5. Write difference between algorithm and flowchart.[U][CO2]
6. Write an algorithm and draw flowchart to find the sum of numbers from 1 to n[A][CO2]
7. Write an algorithm and draw flowchart to find whether the given number is prime or not.[A][CO2]
8. Define compiler, interpreter, assembler, linker and loader. [R][CO1]
9. Explain the importance of C language.[U][CO1]
10. Explain the structure of C program and explain[U][CO1]
11. What is format specifier? [U][CO1]
12. Define keyword, constant and variable. [U][CO1]
13. Explain different constants in C.[R][CO1]
14. What are different types of type casting? [U][CO1]
15. Explain sizeof() with example? [U][CO1]
16. Why do we use header files? [U][CO1]
17. Define relational operator? [R][CO1]
18. What is the purpose of adding comments in a program? [U][CO1]
19. Differentiate between computer software and hardware?[U][CO1]

20. Write detailed notes on C data types. [U][CO1]

21. Explain the difference between pre increment and post increment operator with appropriate examples [U][CO1]

22. Write about conditional operator (ternary operator). [U][CO1]

23. Discuss about the following operators in C language with example. [U][CO2]

a. Bitwise operators

b. Relational operators

c. Logical

operators

24. Perform the following operations [A][CO2]

a.  $23 \gg 3$

b.  $27 \ll 2$

c.  $15 \& 9$

d.  $15 \wedge 9$

e.  $15 | 9$

25. Write a program in C to perform swapping of two numbers without using temporary variable. [A][CO2]

### **Example Questions from UNIT-1 MCQ's**

#### **PROGRAMMING FOR PROBLEM SOLVING**

#### **UNIT-I**

#### **OBJECTIVE QUESTIONS**

1. The smallest unit of data in computer is \_\_\_\_\_ [ R ] [CO1]

- a) Byte b) Nibble c) Bit d) KB ----

Answer: c

Explanation: A bit is defined as the smallest unit of data in a computer system. It is used as a short form of Binary Digit. A bit can have only two values 0 or 1. A nibble comprises of 4 bits; a byte is a collection of 8 bits whereas KB (Kilobyte) is equal to 1024 bytes.

2. The only language which the computer understands is \_\_\_\_\_ [ U ][CO1]

- a) Assembly Language b) Binary Language c) BASIC d) C Language

Answer: b

Explanation: The Computer understands only binary language which is written in the form of 0s & 1s. A computer can understand assembly language but an assembler is required which converts the assembly language to binary language. Similarly, for understanding high level languages, compilers/interpreters are required.

3. Which of the following is incorrect? [ U ] [CO2]

Algorithms can be represented:

a) as pseudo codes

b) as syntax

c) as programs

d) as flowcharts

Answer: b

Explanation: Representation of algorithms: -As programs -As flowcharts -As pseudo codes.

4. Interpreter is used as a translator for \_\_\_\_\_ [U][CO1]

a) Low level language b) High Level Language c) COBOL d) C

Answer: b Explanation: It is generally used to make the code into a machine understandable format. Interpreter is used with the high-level languages similarly. Assembler is used in case of low-level languages.

5. What do you call a specific instruction designed to do a task? [U][CO1]

a) Command b) Process c) Task d) Instruction

Answer: a

Explanation: A program is a set of instructions. A command is given to do a specific job. A program in execution is called a process.

6. Choose correct answer based on the following C program structure [A][CO2]

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

```
main ()
```

```
{
```

```
/* Multi Line Comment
```

```
This line is ignored by compiler
```

```
*/
```

```
print("Hello C..");
```

```
}
```

A) #include is a Preprocessor Directive

B) <stdio.h> is a header file with predefined functions like printf, scanf etc

C)

```
#include
```

```
main()
```

```
{  
}
```

is a mandatory function to be included in every C Program.

- D) All the above

Answer : D

Explanation: every C program need to have above lines of code.

7. An Identifier may contain?

[U][CO1]

- A) Letters a-z, A-Z in Basic character set.
- B) Underscore \_ symbol
- C) Numbers 0 to 9
- D) All the above

Answer: D

Explanation: an identifier contains letters, numbers and \_ symbol.

8. Choose correct statements

[U][CO1]

- A) A constant value does not change.

A variable value can change according to needs.

- B) A constant can change its values.

A variable can have one constant value only.

- C) There is no restriction on number of values for constants or variables.

- D) Constants and Variables cannot be used in a single main function.

Answer: A

Explanation:

Constant value is always constant. Constant is also called Literal.

Variable can have any number of arbitrary values and once value at any point of time.

Variable is also called Identifier.

9. Number of Keywords present in C Language are.? [R][CO1]

A) 32

B) 34

C) 62

D) 64

Answer: A

Explanation:

Only 32 Keywords originally. Compilers are individual companies can include and use extra keywords if required. Such keywords should precede with \_\_ (two Underscore symbols before names).

10. What will be the output of following program?

[A][CO2]

```
#include <stdio.h>

void main()
{
    int x=(20 || 40 ) && (10);
    printf("x= %d",x);
}
```

a. x= 60

b. x= 70

c. x= 0

d. x= 1

Answer: d

1)(20 || 40) .... both are non-zero values, will return 1.

2)(1) && 10 .... both are non-zero values; hence output will be 1.

11. Associativity of C Operators \*, /, %, +, - and = is.?

[A][CO2]

A) Operators \*, / and % have Left to Right Associativity. Operators + and - have Left to Right Associativity. Operator = has Right to Left Associativity.

B) Operators \*, / and % have Right to Left Associativity. Operators + and - have Left to Right Associativity. Operator = has Right to Left Associativity.

C) Operators \*, / and % have Right to Left Associativity. Operators + and - have Right to Left Associativity. Operator = has Right to Left Associativity.

D) Operators \*, / and % have Right to Left Associativity. Operators + and - have Right to Left Associativity. Operator = has Left to Right Associativity.

Answer: A

Explanation:

Operators \*, / and % have Left to Right Associativity. Operators + and - have Left to Right Associativity. Operator = has Right to Left Associativity.

12. What will be the output of following program? [A][CO2]

```
#include <stdio.h>

int main(){

    int x;

    x=100,30,50;

    printf("x=%d\n",x);

    x=(100,30,50);

    printf("x=%d\n",x);

    return 0;

}
```

A) x=100 x=100    B) x=100 x=50 C) x=50 x=50 D) x=50 x=100

Answer: B

Since = (assignment operator) has more precedence than comma operator (,), so = operator evaluates first and 100 will be assigned to x.

In second case, x= (100,30,50), here () have more precedence so (100,30,50) will be evaluated first from left to right, and x will be 50.

13. Left shift (<<) and Right shift (>>) operators are equivalent to \_\_\_\_\_ and \_\_\_\_\_ by 2.

[U][CO1]

- A) Multiplication and Division
- B) Division and Multiplication
- C) Multiplication and Remainder

## D) Remainder and Multiplication

Answer: A

Explanation: Left shift by 1 return the multiplication by 2 and Right shift by 1 return the division by 2.

14. What is the output of following program?

[A][CO1]

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

```
int main()
{
    int a = 1;
    int b = 1;
    int c = a || --b;
    int d = a-- && --b;
    printf("a = %d, b = %d, c = %d, d = %d", a, b, c, d);
    return 0;
}
```

- (A) a = 0, b = 1, c = 1, d = 0
- (B) a = 0, b = 0, c = 1, d = 0
- (C) a = 1, b = 1, c = 1, d = 1
- (D) a = 0, b = 0, c = 0, d = 0

Answer: (B)

Explanation: Let us understand the execution line by line.

Initial values of a and b are 1.

Since a is 1, the expression --b is not executed because of the short-circuit property of logical or operator So c becomes 1, a and b remain 1

```
int c = a || --b;
```

The post decrement operator -- returns the old value in current expression and then updates the value. So the value of expression a-- is 1. Since the first operand of logical and is 1, shortcircuiting doesn't happen here. So

the expression --b is executed and --b returns 0 because it is pre-increment.

The values of a and b become 0, and the value of d also becomes 0.

15. What will be output of the following program?

[A][CO1]

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

```
int main(){
```

```
    int i=1;
```

```
    i=2+2*i++;
```

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

```
    return 0;
```

```
}
```

A)5

B)4

C)6

D)3

Explanation:

i++ i.e. when postfix increment operator is used any expression the it first assign the its value in the expression the it increments the value of variable by one. So,

$i = 2 + 2 * 1$

$i = 4$

Now i will be incremented by one so  $i = 4 + 1 = 5$

16.What will be output of the following program?

[A][CO1]

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

```
void main(){
```

```
    int x;
```

```
    x=10,20,30;
```

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

```
    return 0;
```

```
}
```

B) 9    A)10

C) 8

D) 11

Explanation :

Since assignment operator (=) has more precedence than comma operator .So = operator will be evaluated first than comma operator. In the following expression

x = 10, 20, 30

First 10 will be assigned to x then comma operator will be evaluated.

17. What is the only function all C programs must contain? [R][CO1]

- (a) Start()
- (b) System()
- (c) Main()
- (d) Program()
- (e) None of the above.

Answer: C

Explanation : Since main() is the starting of a program execution.

18. Which of the following is a correct statement? [U][CO1]

- (A) Variable name must start with underscore
- (B) Variable name must have digit
- (C) Variable name must have white space character
- (D) Keyword cannot be a variable name

Answer: D

Explanation: in program we cant use keyword as variable because they are predefined .

19. What is algorithm? [U][CO1]

- a) Application code
- b) Type of programming language
- c) a step

Answer:d

Explanation: an algorithm is a step by step procedure to solve a problem

20. Which part of CPU perform calculations and make decision [R][CO1]

- A) Alternate Logic Unit
- B) Arithmetic Logic Unit
- C) Arithmetic Local Unit
- D) Alternate Local Unit

Answer: B

Explanation: ALU is one of the major components which perform calculations



**Course Code: 20CS C01**

**Course Title: PROGRAMMING FOR PROBLEM SOLVING**

**Example Questions from UNIT-2 MCQ's**

**Short answer questions**

1. What are the control structures available in C(L1)(CO3)  
If, if else, nested if else, switch, for loop, while loop, do while loop, break, continue and goto
2. Name the selective constructs in c(L1)(CO3)  
If, if else, nested if else, else if ladder, switch
3. Distinguish between a pre test looping and post test looping(L4)(CO3)
4. Name the loop constructs in c. (L1)(CO3)
5. Name the unconditional control statements in C(L1)(CO3)
6. Name the conditional control structures in C. (L1)(CO3)
7. Explain in detail the different forms of **if** constructs in c. (L2)(CO3)
8. What is conditional expression(L1)(CO3)
9. Illustrate the use of ternary operations. (L2)(CO3)
10. Explain the **while** structure with examples(L2)(CO3)
11. Explain the **do-while** structure with examples(L2)(CO3)
12. Can the action of a **do-while** structure is simulated by **if** or **if-else** structures? (L4)(CO3)
13. Explain with an example.(L2)(CO3)
14. Compare the **while** structure with **do-while** structure
15. Write the advantages of **while** loop over the **do-while loop**(L2)(CO3)
16. How is a for loop equivalently written using a while loop.(L2)(CO3)
17. Under what circumstances will the do-while be more appropriate than the for loop.(L2)(CO3)
18. What is a comma expression?(L2)(CO3)
  19. What is the purpose of continue statement?(L2)(CO3)
  20. What is the propose of break statement?(L2)(CO3)
  21. Differentiate between break and continue statements.(L4)(CO3)
22. What is meant by nested loop?(L2)(CO3)
23. Explain switch-case structure with example.(L2)(CO3)
24. Compare the switch-case with if else structure.(L4)(CO3)
25. Why goto statements are discouraged?(L2)(CO3)

26. Rewrite the following without using compound relations(L2)(CO3)

```
if(EM>90&&EG>95||Total>190)
    printf("SUCCESS");
else
    printf("Failure");
```

26. what is the output of the following program(L2)(CO3)

```
main()
{
    int i;
    for(;i;)
        i--;
    printf("%d",i);
}
```

27. Distinguish between source code, object code and executable code.(L4)(CO3)

28. Draw flow chart for various loops.(L3)(CO3)

29. Draw flow chart for various control constructs.(L3)(CO3)

## Programs:

1. Develop a program to find sum of digits of a given integer number.(L3)(CO3)
2. Develop a program to print the given integer number in reverse order.(L3)(CO3)
3. Develop a program to check whether the given number is palindrome or not.(L3)(CO3)
4. Develop a program to find largest of given two numbers.(L3)(CO3)
5. Develop a program find largest of given three numbers by using simple if, nested if, else if ladder(L3)(CO3)
6. Develop a program to check whether the given triangle is right triangle or not.(L3)(CO3)
7. Develop a program to find the quadrant position of given coordinate.(L3)(CO3)
8. Develop a program to check whether the entered character is vowel or not.(L3)(CO3)
9. Develop a program to print CBIT twenty times by using for loop.(L3)(CO3)
10. Develop a program to find the sum of first n natural numbers.(L3)(CO3)
11. Develop a program to find the sum of first n even numbers.(L3)(CO3)
12. Develop a program to find the sum of given n numbers.(L3)(CO3)
13. Develop a program to print n<sup>th</sup> table upto 10.(L3)(CO3)
14. Develop a program to print the following triangle of numbers by using nested loops.(L3)(CO3)

```
1
1 2
1 2 3
1 2 3 4
```

```
1
1 2
1 2 3
1 2
1
```

```
1
1 2
```

|   |   |   |   |   |
|---|---|---|---|---|
| 1 | 2 | 3 |   |   |
|   | 1 |   |   |   |
| 1 | 2 | 1 |   |   |
| 1 | 2 | 3 | 2 | 1 |
|   | 1 |   |   |   |
| 2 | 1 | 2 |   |   |
| 3 | 2 | 1 | 2 | 3 |

- 15.**Write a program to print the sum of first n terms of sine series.(L3)(CO3)  
**16.**Write a program to print the sum of first n terms of cosine series.(L3)(CO3)  
**17.**Write a program to print the Pascal triangle.(L6)(CO3)  
**18.**Write a program to find  $n^{\text{th}}$  Fibonacci number.(L3)(CO3)  
**19.**Write a program to generate first n terms of Fibonacci series.(L3)(CO3)  
**20.**Write a program to find factorial of a number.(L3)(CO3)  
**21.**Write a program to check whether the given number is prime or not.(L3)(CO3)  
**22.**Write a program to print first n prime numbers.(L3)(CO3)  
**23.**Write a program to print prime numbers upto n.(L3)(CO3)  
**24.**Write a program to print prime numbers between n1 and n2.(L3)(CO3)  
**25.**WAP to display menu for selecting addition, subtraction, multiplication and division and compute according to menu chosen. (Use do-while).(L3)(CO3)  
**26.**WAP to addition, subtraction, and multiplication of two complex numbers  $(a+ib)$  and  $(x+iy)$ .(L3)(CO3)  
**27.**Using while loop, write a program to print all Armstrong numbers between 1 and 500. If sum of cubes of each digit of the numbers is equal to the number itself, then number is called Armstrong number.(L3)(CO3)  
For example:  $153 = (1*1*1) + (5*5*5) + (3*3*3)$   
**28.**WAP to find whether the given number is strong or not using functions.(L3)(CO3)  
(Example:  $145 = 1! + 4! + 5! = 1 + 24 + 120 = 145$  )  
**29.**Input a year from keyboard and determine whether it is leap year or not using the conditional operator.(L3)(CO3)  
**30.**Accept three numbers from user and determine which of the two numbers or all three numbers are equal or not using logical operators.(L3)(CO3)  
**31.**WAP to calculate  $nPr = n!/(n-r)!$ (L3)(CO3)  
**32.**WAP to calculate  $nCr = n! / n!*(n-r)!$ (L3)(CO3)  
**33.**WAP to find whether the given number is perfect or not  
Example ( 6 is perfect number since  $6 = 1 + 2 + 3$  )(L3)(CO3)  
**34.**WAP to convert given decimal number to binary number by using a function.(L3)(CO3)

## **MCQS:**

### **Review and revision:**

#### **MCQs:**

1. In C, parameters are always(L2)(CO4)
  - a) Passed by value
  - b) Passed by reference
  - c) Non-pointer variables are passed by value and pointers are passed by reference
  - d) Passed by value result

**Answer: a**

2. A function prototype is used for (L2) (CO4)
  - a) Declaring the function logic
  - b) Calling the function from the main body
  - c) Telling the compiler, the kind of arguments used in the function
  - d) Telling the user for proper use of syntax while calling the function

**Answer: c**

3. The variables that are declared inside a function are (L1) (CO4)
  - a) global variable
  - b) local variables
  - c) static variables
  - d) must be same as the variables declared in main().

**Answer: b**

- 4.

What is the output of the following C program?

```
#include <stdio.h>
void foo(), f();
int main()
{
    f();
    return 0;
}
void foo()
{
    printf("2 ");
}
void f()
{
    printf("1 ");
    foo();
}
```

(L3) (CO4)

- a) Compiler error as foo() is not declared in main
- b) 1 2
- c) 2 1
- d) Compile time error due to declaration of functions inside main

**Answer:** b

5. Can we use a function as a parameter of another function? [Eg: void func1(int func2())]? (L1) (CO4)
- a) Yes, and we can use the function value conveniently
  - b) Yes, but we call the function again to get the value, not as convenient as in using variable
  - c) No, C does not support it
  - d) This case is compiler dependent

**Answer:** c

6. What is the return-type of the function sqrt().(L2) (CO4)
- a) int
  - b) double
  - c) float
  - d) depends on compiler

**Answer:** b

7. What is the default return type if it is not specified in function definition??(L3) (CO4)
- a) void
  - b) integer
  - c) double
  - d) float

**Answer:** b

- 8.

**Identify What is the error in the following program**

```
#include<stdio.h>
int f(int a)
{
    a > 20? return(10): return(20);
}
int main()
{
    int b;
    b = f(20);
    printf("%d\n", b);
    return 0;
}
```

(L3) (CO4)

- a) Error: Return statement cannot be used with conditional operators
- b) Error: Prototype declaration
- c) Error: Two return statements cannot be used in any function
- d) No error

**Answer: a**

**9. Identify which statement is correct about Passing by value parameters.(L3)(CO4)**

- a) It cannot change the actual parameter value
- b) It can change the actual parameter value
- c) Parameters is always in read-write mode
- d) None of them

**Answer: a**

**10. Which keyword is used to give back the value?(L5) (CO4)**

- a) static
- b) void
- c) return
- d) const

**Answer: b**

**11. Evaluate the output of the following program. (L5) (CO4)**

```
#include <stdio.h>
void foo()
{
    return 1;
}
void main()
{
    int x = 0;
    x = foo();
    printf("%d", x);
}
```

- a)Compile time error
- b)0
- c)1
- d) run time error

**Answer: a**

**12. Number of values a functionin C can return? (L2) (CO4)**

- a)2
- b)0
- c)1
- d) 3

**Answer: c**

**13. Types of functions in C language(L2) (CO4)**

- a)User defined
- b)Library
- c)both (a) and (b)
- d) None

**Answer: c**

**14. C program must contain at least(L2) (CO4)**

- a) n functions
- b)1 function
- c) 2 functions
- d) None

**Answer: b**

**15. which of the following is not a storage class specifier?(L2) (CO4)**

- a) auto
- b) register
- c) extern
- d) volatile

**Answer: d**

**16. what is the initial value of register storage class specifier? (L2) (CO4)**

- a)0
- b)garbage
- c)1
- d)infinite

**Answer: b**

**Review and revision exercise (2) :**

**Case Study:**

1. To display the given decimal number in words using user defined function
2. To display the given decimal number in roman numbers using user defined function
3. To find the day name when date is entered using user defined function

**Expected Questions:**

**Short answer type:**

1. What is a function? What is the need for functions?(L1) (CO4)
2. Classify different types of functions. (L2) (CO4)
3. Explain the general form of defining a function. (L2) (CO4)
4. Define i) function prototype ii) function header iii) recursive function (L1) (CO4)
5. Differentiate between (L2) (CO4)
  - i) actual and formal parameters
  - ii) calling function and called function
  - iii) user definid function and library function
  - iv) call by value and call by reference
  - v) function prototype and function header
  - vi) local and global variables
6. Write about storage classes. (L1) (CO4)
7. What are different types of variables? Explain with examples. (L1) (CO4)

**Long answer type:**

8. Develop a C program to find factorial of a number? (L3) (CO4)
9. Develop a C program to find nth Fibonacci number? (L3) (CO4)
10. Develop a C program to find GCD of two numbers? (L3) (CO4)
11. Develop a C program to find sum of digits of a number? (L3) (CO4)
12. Develop a C program to find prime or not? (L3) (CO4)
13. Develop a C program to find palindrome or not? (L3) (CO4)
14. Develop a C program to find Fibonacci series? (L3) (CO4)
15. Develop a C program to find power of a number by using non recursive and recursive functions? (L3)(CO4)



**CHAITANYA BHARATHI  
INSTITUTE OF TECHNOLOGY (A)**

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. [www.cbit.ac.in](http://www.cbit.ac.in)



COMMITTED TO  
RESEARCH,  
INNOVATION AND  
EDUCATION

**41**  
years

**Course Code: 20CS C01**

**Course Title: PROGRAMMING FOR PROBLEM SOLVING**

### **UNIT-III**

#### **Review and revision:**

#### **Review and revision exercise (1) :**

#### **Lecture topic quiz / MCQs: (All these quiz's belongs to BL3 & BL4)**

1. Test for the output of the following program, if the array begins at address 1200?

```
main()
{
    intarr[] = {2,3,4,1,6};
    printf("%d %d",arr, sizeof(arr));
}
```
2. Identify that the array name gives the base address in all the contexts?
3. Test for the output of the following program, if the array begins at address 65486 ?

```
main()
{
    intarr[] = {12,14,15,23,45};
    printf("%u %u",arr, &arr);
}
```
4. Experiment the expressions arr and &arr gives same for an array of 10 integers ? (arr is the array variable name)
5. Test for the output of the following program, if the array begins at address 65486 ?

```
main()
```

```
{  
    intarr[] = {12,14,15,23,45};  
    printf("%u %u",arr + 1, &arr + 1);  
}
```

6. Test for the output of the following program ?

```
main()  
{  
    float a[] = {12.4,2.3,4.5,6.7};  
    printf("\n %d",sizeof(a) / sizeof(a[0]));  
}
```

7. Test for the output of the following program if the array begins at 65472?

```
main()  
{  
    int a[3][4] = {  
        1,2,3,4,  
        4,3,2,1,  
        7,8,9,0  
    };  
    printf("\n %u %u",a + 1, &a + 1);  
}
```

8. Inspect if we pass the name of a 1-D int array to a function it decays into a pointer to an int. If we pass the name of a 2-D array of integers to a function what would it decay into?

#### Multiple Choice Questions

(All MCQ's belongs to (CO3,4,5))

1. An array \_\_\_\_\_ (BL1)
  - a) Is collection of similar data items
  - b) stored data in the continuous memory allocation
  - c) elements share a common name
  - d) **All of the above**
2. What will be the values of 8<sup>th</sup> element in the array if this is the declaration and initialization of the integer array int a[10]={0};(BL1)
  - a) 0
  - b) Garbage Value
  - c) Value depends on compiler
  - d) Compilation error
3. Pick the correct declaration and initialization of 2 dimensional integer array(BL1)
  - a) **int a[][]={1,2,2,4};**
  - b) int a[3][] ={1,2,2,4};;
  - c) int []a[]={1,2,2,4};
  - d) **int a[][3]={1,2,2,4};**
4. Index of the first element in the arrays is \_\_\_\_\_(BL1)
  - a) 0
  - b) -1
  - c) 1
  - d) 2
5. printf("%d",a[3][4]); In this statement which element is accessed?(BL1)
  - a) 4<sup>th</sup> element in the 3<sup>rd</sup> row of the matrix

- b) 3<sup>rd</sup> element in the 4<sup>th</sup> row of the matrix  
 c) 5<sup>th</sup> element in the 4<sup>th</sup> row of the matrix  
 d) 4<sup>th</sup> element in the 5<sup>th</sup> row of the matrix

6. Elements in an array can be accessed\_\_\_(BL2)

- a) Sequentially  
 b) Randomly  
 c) Both  
 d) None  
 e)

7. We want to access the 8<sup>th</sup> element in a one dimensional array int a[50] . which statement is correct?

- a) printf("%d", a[8]);  
**b) printf("%d", a[7]);**  
 c) printf("%d", a[6]);  
 d) printf("%d", a[5]);

8. In the binary search if the key exist in the middle of the array. How many comparison need to be done. (BL2)

- a) 1  
 b) n/2  
 c) n  
 d) 0

9. In the linear search if the key exist in the middle of the array. How many comparison need to be done. (BL2)

- a) 1  
**b) n/2**  
 c) n  
 d) 0

10. Consider a two dimensional array int a[5][5]. Assume each element occupies 2 bytes. The base address of the array is 1000, elements are stores in row major order. What is the address of a[3][2]?

- a) 1034(BL3)**  
 b) 1030  
 c) 1036  
 d) 1032

11. The maximum dimensional array variable that can be declared in C. (BL2)

- a) 1  
 b) 2  
 c) 3  
**d) No limit**

12. Pick the operations which cannot be done on an array variable 'a' (BL2)

- a) ++a, a++  
 b) a+1, 1+a, a=a+1, a+=1  
 c) a[i]++, i[a]++, \*(a+i)++, \*(i+a)++  
 i. **both a and b are not allowed**  
 ii. both a and c are not allowed  
 iii. both c and b are not allowed  
 iv. none

13. Pick the uses of arrays (BL2)

- a) No need to declare many separate variable to handle many data items  
 b) Using single variable and subscript all the data items can be accessed  
 c) Maintaining the code is easy and remembering of variables is reduced on programmers side

**d) All**

14. Pick the uses of multidimensional arrays (BL2)
- Able to store data in rows and columns
  - It is an array of arrays
  - In all the dimensional the index will start from '0' and ends at 'sizeofthedimension-1'
  - All
15. In CBIT in the academic year 2020-2021 first year contains 20 sections, in each section 60 students exist, each student is having 6 courses. The secured by all students in all subjects in 20 sections need to be stored. Pick the suitable multidimensional variable to be used to store the data. If data to be stored in row major order of sections, students and subjects (BL3)
- Marks[20][60[6];]
  - Marks[60][20[6];]
  - Marks[20][6[60];]
  - Marks[6][60[20];]

**Review and revision exercise (2) :**

- Write a program to find the determinant of a matrix. (CO3,4,5)
- Write a program to count number of duplicate elements exist in an array.(CO3,4,5)
- Write a program to display each element of the array in words.(CO3,4,5)
- Write a program to display array elements in pyramid form.(CO3,4,5)

Example: 1 2 3 4 5 6 are array elements. Then output should be

```
    1  
   2     3  
  4       5       6
```

- Write a program to find the inverse of a square matrix.(CO3,4,5)
- Write a program to perform the below operations on elements of a matrix(CO3,4,5)
  - Find the minimum and maximum element
  - Sort them in ascending order and store in another one dimensional variable
  - Count how many zeros exist in the matrix
  - Count how many duplicates exist
  - Swap any two elements in the matrix based on user choice

**Discussion questions / Expected Questions:**

**Questions under Blooms Level-1**

- List out the advantages of an array. (CO5)
- Define an array. Declare an array which holds average of 60 students.(CO5)
- What is the significance of index in an array. (CO5)
- int a[n]; does this declaration is correct? Justify your answer. (CO5)
- Outline the significance of array name. (CO5)
- What is the significance of type and size in the array declaration? (CO5)

**Questions under Blooms Level-6**

- Develop a program for sorting n integers using selection sort/bubble sort. (CO3,4,5)
- Distinguish Lvalue and Rvalue of an array element. Explain the difference with example. (CO5)
- Develop a program to store 'n' terms of a Fibonacci series in a variable length array. (CO3,4,5)
- Discuss binary search/linear search with appropriate example in detail. (CO5)
- Discuss bubble sort/selection sort with appropriate example in detail. (CO5)

**Questions under Blooms Level-2,3**

12. Explain in detail about the initialization, access and memory allocation of one dimensional arrays with the help of a diagram. (CO5)
13. Explain in detail about the initialization, access and memory allocation of two dimensional arrays with the help of a diagram. (CO5)
14. Compare a scalar variable and arrays. (CO5)
15. Demonstrate with an example about passing an array to a function.(CO3,4,5)
16. Illustrate with an example about variable length array.(CO5)
17. An integer array is declared of size 10. If the first element of the array is stored at 0x12345 memory location. Find the memory location where 6<sup>th</sup> element of the array will be. (CO5)

**Review and revision:**

**MCQs:**

1. Which function will you choose to join two words?(L1)(CO4)
  - a) strcpy()
  - b) strcat()
  - c) strncon()
  - d) memcon()
 Answer: b
2. Show the function that appends not more than n characters.(L2)(CO4)
  - a) strcat()
  - b) strcon()
  - c) strncat()
  - d) memcat()
 Answer: c
3. Which of the following is the variable type defined in header <string.h>?(L1)(CO3)
  - a) sizet
  - b) size
  - c) size\_t
  - d) size-t
 Answer: c
4. Identify whether NULL is the macro defined in the header <string.h>.(L3)(CO5)
  - a) true
  - b) false
 Answer: a
5. Recall is there any function declared as strstr()?(L1)(CO4)
  - a) true
  - b) false
 Answer: a
6. Analyze which of the following function returns a pointer to the located string or a null pointer if string is not found.(L4)(CO4)
  - a) strtok()
  - b) strstr()
  - c) strspn()
  - d) strrchr()
 Answer: b
7. Identify which of the given function is used to return a pointer to the located character?(L3)(CO4)
  - a) strrchr()

b) strxfrm()

c) memchar()

d) strchr()

Answer: d

8. Analyze what will be the output of the following C code?(L4)(CO5)

```
char str1[] = "Helloworld ";
char str2[] = "Hello";
intlen = strspn(str1, str2);
printf("Length of initial segment matching %d\n", len );
```

a) 6

b) 5

c) 4

d) no match

Answer: b

9. Identify thefunction that returns the number of characters that are present before the terminating null character.(L3)(CO4)

a) strlength()

b) strlen()

c) strlent()

d) strchr()

Answer: b

10. Examine the output of the following C code?(L4)(CO5)

```
char str1[15];
char str2[15];
int mat;
strcpy(str1, "abcdef");
strcpy(str2, "ABCDEF");
mat= strncmp(str1, str2, 4);
if(mat< 0)
printf("str1 is not greater than str2");
else if(mat> 0)
printf("str2 is is not greater than str1");
else
printf("both are equal");
```

a) str1 is not greater than str2

b) str2 is not greater than str1

c) both are equal

d) error in given code

Answer: b

11. Which function tests for any character for which isalpha or isdigit is true.(L1)(CO4)

a) isxdigit()

b) isspace()

c) isalnum()

d) isupper()

Answer: c

12. Which function returns true only for the characters defined as lowercase letters?(L1)(CO4)

a) islow()

b) islower()

c) isalpa()

d) isalnum()

Answer: b

**Review and revision exercise (2) :**

**Case Study:**

WAP to implement Timetable and Invigilation chart with the following options:

1. Class Timetable generation  
(Input: Number of Theory and Lab subjects, No. of faculty and their names, each faculty workload ) theory – 3hrs/week, lab-4hrs/week(split of 2hrs)
2. Faculty Timetable generation  
(Input: faculty ID)
3. Faculty free hours available  
(Input: faculty ID)
4. Invigilation Chart generation  
(Input: No. of days of examination, number of invigilators required per day)

**Expected Questions:**

Short answer type:

17. Why do we have a null character ('\0' or NUL) at the end of a string? (L1)(CO5)
18. Apply character manipulation functions and develop a C program that will capitalize all the letters of a string. (L3,L6)(CO4)

Long answer type:

19. Analyze appropriate string manipulation functions and develop a C program that deletes a word from a sentence. Note that the word may appear any number of times.(L4,L3)(CO4)
20. Develop a C program that reads a line of text and counts all occurrences of a particular word.(L6)(CO5)
21. Explain any four string manipulation functions from <string.h> library with examples. (L2)(CO4)



**Course Code: 20CS C01**

**Course Title: PROGRAMMING FOR PROBLEM SOLVING**

### **UNIT-IV**

#### **1.Objective Questions /Multiple Choice questions :**

1.(\*FP)(INT, INT) REPRESENTS .....

L2, CO5

**A. FUNCTION POINTER**

B. ARRAY OF PINTERS

C. POINTER ARRAY

D. ARRAY OF FUNCTION POINTER

2.\_\_\_\_\_ IS THE MEMORY VARIABLE USED TO STORE ADDRESS OF ANOTHER VARIABLE.

L1, CO5

A. FUNCTION

B. VARIABLE

**C. POINTER**

D. IDENTIFIER

3.INT \*P;

L3, CO5

INT A=100;

P=&A;

PRINTF("%d %d",\*P,P);

**A.100,1000**

B. 10,1000

C. 10

D. 1000

4.Address stored in pointer variable is of \_\_\_\_\_ type

L1,CO5

**A. Integer**

B.character

C. Float

D.Double

5.\* is called as \_\_\_\_\_

L1,CO5

### **A.Value at pointer**

- B.Address operator
- C.Scope resolution operator
- D.None

6.int \*p1,\*p2; find out valid statement L2,CO5

A.p1-p2

**B.p1\*p2**

C.p1+p2

D.p1/p2

7.The operator used to get value at address stored in a pointer variable is..... L1,CO5

A.\*

B.&

C.&&

D.||

8. What would be the equivalent pointer expression for referring the array element a[i][j][k][l]. L3, CO5

A.(((a+i)+j)+k)+l)

**B.\*(\*(\*(a+i)+j)+k)+l)**

C.((a+i)+j)+k+l)

D.((a+i)+j+k+l)

9.A pointer is ..... L1, CO5

A.A keyword used to create variables

B.A variable that stores address of an instruction

**C.A variable that stores address of other variable**

D.All of the above

10.What will be the output? L4, CO5

```
main()
{
char *p;
printf("%d %d",sizeof(*p),sizeof(p));
```

A.1 1

**B.1 2**

C.2 1

D.2 2

11.Which of the following statements correct about k used in the below statement?

```
char ****k; L3, CO5
```

A.k is a pointer to a pointer to a pointer to a char

**B.k is a pointer to a pointer to a pointer to a pointer to a char**

C.k is a pointer to a char pointer

D.k is a pointer to a pointer to a char

12.main()

L4, CO5

```
{  
char *p;  
p="Hello";  
printf("%c\n", *p);  
}
```

**Answer: H**

13. Multiple indirection operator is \_\_\_\_\_

L1, CO5

- A.-->
- B.&
- C.\*
- D.\*\***

14. A pointer to pointer points to the address of a

L1, CO5

- A. Structure
- B. Union
- C. Array
- D. Pointer**

15. Which is the correct way to declare a pointer?

L2, CO5

- A. int \*ptr;
- B. int \* ptr;
- C. int\* ptr;
- D. All**

16. Prior to using a pointer

L1, CO5

- A. it should be declared
- B. it should be initialized
- C. it should be declared and initialized**
- D. None

17. int k[3]={1,2,3};

L2, CO5

int \*p;

one of the following statement is equal to p=k is

- A. p=&k[0]**
- B. p=&k[1]
- C. p=&k[2]
- D. None

18. How to combine the following two statements into one?

L2, CO5

```
char *p; p=(char*)malloc(100);
```

- A. char p=malloc(100);
- B. char \*p=(char)malloc(100);
- C. char \*p=(char\*)malloc(100);**
- D. char \*p=(char\*)(malloc\*)(100);

19.If the size of integer is 4 bytes,

L5, CO5

what will be the output of the program?

```
int main()
{
int arr[]={12,13,14,15,16};
printf("%d, %d, %d\n",sizeof(arr),sizeof(*arr),sizeof(arr[0]));
return 0;
}
A.10, 2, 4
B.20, 4, 4
C.16, 2, 2
D.20, 2, 2
```

20.What will be the output?

L5, CO5

```
main()
{
char *p;
printf("%d %d",sizeof(*p),sizeof(p));
}
A.1 1
B.1 2
C.2 1
D.2 2 20.
```

21. What will be the output?

L5, CO5

```
main()
{
printf"%d %d",sizeof(int *),sizeof(int **));
}
A.4 4
B.0 2
C.2 2
D.2 4
```

22.Pointers are of

L1, CO5

- A. integer data type**
- B.character data type
- C.unsigned integer data type
- D.none of these

23.main()

L5, CO5

```
{
char *str1="abcd";
char str2[]={abcd";
printf("%d %d %d",sizeof(str1),sizeof(str2),sizeof("abcd")); }
```

**Answer: 2 5 5**

24. main()

L3, CO5

```

{
int *j;
{
int i=10;
j=&i;
}
printf("%d",*j);
}

```

**Answer: 10**

```

25.main()
{
char *p;
int *q;
long *r;
p=q=r=0; p
++;
q++;
r++;
printf("%p...%p...%p",p,q,r);
}

```

**Answer: 0001...0002...0004**

## 2. Short Questions :

- |                                                           |         |
|-----------------------------------------------------------|---------|
| 1. Define pointer. How can you declare it?                | L2, CO5 |
| 2. What is pointer arithmetic?                            | L2, CO5 |
| 3. Define pointer array.                                  | L1, CO5 |
| 4. What is pointer to pointer?                            | L1, CO5 |
| 5. Define Void Pointer.                                   | L1, CO5 |
| 6. Define Null Pointer.                                   | L1, CO5 |
| 7. Write about operators used with pointers.              | L2, CO5 |
| 8. Differentiate Pointers and Arrays.                     | L3, CO5 |
| 9. Write a function to swap two variables using pointers. | L3, CO5 |
| 10. Differentiate malloc( ) and calloc( ).                | L2, CO5 |

## Long Questions :

- |                                                                                                              |         |
|--------------------------------------------------------------------------------------------------------------|---------|
| 1. Write a program to calculate length of the string using pointers.                                         | L6, CO5 |
| 2. Write a C program to illustrate the use of indirection operator to access the value pointed by a pointer. | L6, CO5 |
| 3. What are the features of pointers? Write a C program to print address of a variable.                      | L3, CO5 |
| 4. Explain the declaration of pointers and pointer to pointer with examples.                                 | L4, CO5 |
| 5. Explain the concept of array of pointers with examples.                                                   | L3, CO5 |
| 6. With proper examples explain different arithmetic operations on pointers.                                 | L3, CO5 |
| 7. Explain the concept of functions returning pointers with example.                                         | L3, CO5 |

8. Write a C program to read and print an array of elements using pointers. L6, CO5
9. Write a C program to read and display multiple strings using pointers. L6, CO5
10. Write a C Program to perform matrix addition using dynamic memory allocation. L6, CO5
11. Write a program to perform matrix multiplication using pointers. L6, CO5
12. Write a program to find the largest in an array using pointers. L6, CO5
13. Write a Program to reverse the string using pointers. L6, CO5
14. Explain the concept of functions returning pointers with example. L3, CO5
15. The roots of a quadratic equation of the form  $ax^2 + bx + c = 0$  are given by the following equations:  $X_1 = -b + \sqrt{b^2 - 4ac} / 2a$ ;  $X_2 = -b - \sqrt{b^2 - 4ac} / 2a$ ; Write a function to calculate the roots. The function must use two pointers, one to receive the coefficients and the other to send the roots to the calling function. L6, CO5
16. Write a C program to arrange the given names in alphabetical order using pointers. L6, CO5
17. Write a C program, using pointer for string comparison. L6, CO5
18. Write a C program to arrange the given numbers in ascending order using pointers. L6, CO5
19. How to use pointer variables in expressions? Explain through examples. L3, CO5
20. What are the various operations performed in pointers explain with an example? L4, CO5

## Structures :

- Which of the following is true for definition of a structure \_\_\_\_\_  
 A) Items of the same datatype      **B) Items of the different datatype**  
 C) Integers with userdefinednames      D) List of Strings      L1
- The keyword used to define a structure is \_\_\_\_\_  
 A) stru      **B) struct**      C) structure      D) STRUC      L1
- The operator used to access the structure member is \_\_\_\_\_  
 A)\*      B)&      C).      D)|      L1
- The operator exclusively used with pointer to structure is \_\_\_\_\_  
 A).      B)[]      **C) →**      D)\*      L1
- Which of the following is correct for a Structuredefinition?  
 A) Scalar data type      **B) Derived data type**  
 B) C) Enumerated type      D) NullType      L1
- When accessing a structure member, the identifier to the left of the dot operator is  
 A) A structure member      B) The structure tag  
**C) A structure variable**      D) The keyword struct      L2
- When a structure is an element to another structure, it is called as a \_\_\_\_\_  
 A) Union      **B) Structure within a structure**  
 C) Pointer to Structure      D) Array of Structures      L1
- A \_\_\_\_\_ structure is one which contains a pointer to its own type.  
 A) Self-referential      B) Nested  
 B) C) Array      D) Pointer      L1
- Consider the following declaration of union st

```
{  
char c;  
int x;  
float y;  
}p;
```

How many bytes are allocated to union variable **p**?

- A) 7bytes    B)**4bytes**    C)1byte    D) 2bytes    L2

10. The size of structure and union is same when they contain \_\_\_\_\_

- A) **Singlember**    B) any number of members  
C) Arrays of different types    D) Pointers to different types    L2

11. The operator used to find the size of any variable \_\_\_\_\_

- A) **sizeof()**    B)sizof()    C) sizeof()    D) size()    L1

12. The operator → is same as the combination of the operators \_\_\_\_\_

- A) \*and .    B) &and.    C) \*and &    D) & and|    L1

13. Union can store \_\_\_\_\_ number of values at a time

- A) All its members    B) **Only 1**  
B) C) 2    D) Cannot hold value    L1

14. ‘C’ provides a facility for user defined datatype using \_\_\_\_\_ concept

- A) Array    B)Function    C) Pointer    D) **Structure** L1

15. In the expression p→ value, p is a

- A) Address    B)**Pointer**    C)Structure    D) Header    L1

16. In C language the expression (\*ps).x is equal to \_\_\_\_\_

- A) ps->x    B)x->ps    C)ps->\*x    D)None    L1

17. Which of the following is a list of named integer constants?

- A) typedef    B)**enumeration**    C)structure    D)union    L1

18. Which of the following is a memory location that is shared by two or more different types of variables?

- A) typedef    B)enumeration    C)structure    D)**union**    L2

19. Which of the following are themselves a collection of different data types?

- A) string    B)**structures**    C) char    D) all of the mentioned    L1

20. User-defined data type can be derived by \_\_\_\_\_

- A) struct    B) enum    C) typedef    D) **all of the mentioned**    L2

21. What is the right way to access value of structure variable book{ price, page }?

- A. printf("%d%d", book.price, book.page);

- B. printf("%d%d", price.book, page.book);

- C. printf("%d%d", price::book, page::book);

- D. printf("%d%d", price->book, page->book);

### **Short Questions :**

- |                                                                                           |    |
|-------------------------------------------------------------------------------------------|----|
| 1. Define Structure? How to Initialize a Structure?                                       | L2 |
| 2. How to represent self-referential structures?                                          | L2 |
| 3. Define Union? How to represent an union?                                               | L2 |
| 4. Write some of the differences between Structure and Union?                             | L1 |
| 5. What are the Different ways of representing Structures and Functions?                  | L2 |
| 6. What is the difference between a structure and a union?                                | L1 |
| 7. What is a member?                                                                      | L1 |
| 8. How is a structure different from an array?                                            | L1 |
| 9. What are member, tag, and variable name in a structure and what purpose do they serve? | L1 |
| 10. What keyword is used in C to create a structure?                                      | L1 |
| 11. What is the difference between a structure tag and a structure instance?              | L2 |

### **Long Questions :**

- |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |    |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| 1. Define Structure and write the general syntax for declaring and accessing members.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | L1 |
| 2. How to copy and compare structure variables? Illustrate with example.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | L2 |
| 3. Write a C program that defines a structure employee containing the details such as empno, empname, department name and salary. The structure has to store 20 employees in an organization. Use the appropriate method to define the above details and define a function that will display the contents?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | L3 |
| 4. Explain the Nested structures                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | L2 |
| 5. Write a C program to read and display student details using structure.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | L3 |
| 6. Define union. Give the general template for union.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | L1 |
| 7. List out the differences between unions, structures and arrays                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | L4 |
| 8. How data elements are stored under unions, explain with example?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | L3 |
| 9. Write a C program to illustrate the concept of structure within structure.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | L3 |
| 10. Declare a structure to store the following information of an employee- <ul style="list-style-type: none"><li>• Employee code</li><li>• Employee name</li><li>• Salary</li><li>• Department number</li><li>• Date of join(it is itself a structure consisting of day, month and year)</li></ul> Write a C program to store the data of ‘n’ employees where n is given by the user (Use dynamic memory allocation). Include a menu that will allow user to select any of the following features: <ol style="list-style-type: none"><li>a. Use a function to display the employee information getting the maximum and minimum salary.</li><li>b. Use a function to display the employee records in ascending order according to their salary.</li><li>c. Use a function to display the employee records in ascending order according to their date of join.</li><li>d. Use a function to display the department wise employee records</li></ol> | L6 |
| 11. Write a program using enumerated types which when given today’s date will print out tomorrow’s date in the form 31st January.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | L3 |
| 12. Write a simple database program that will store a person’s details such as age, date of birth, and address.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | L5 |



**Course Code: 20CS C01**

**Course Title: PROGRAMMING FOR PROBLEM SOLVING**

## UNIT-V

### Question Bank for Unit –V Part 1

#### A. MCQS

1. Choose correct syntax for opening a file. (CO6)(BT1)

- a) FILE \*fopen(const \*filename, const char \*mode)
- b) FILE \*fopen(const \*filename)
- c) FILE \*open(const \*filename, const char \*mode)
- d) FILE open(const\*filename)

Answer: a) FILE \*fopen(const \*filename, const char \*mode)

2. Outline the function of the mode ' w+'? (CO6) (BT 2)

- a) create text file for writing, discard previous contents if any
- b) create text file for update, discard previous contents if any
- c) create text file for writing, do not discard previous contents if any
- d) create text file for update, do not discard previous contents if any

Answer:b) create text file for update, discard previous contents if any

3. choose the correct difference between fscanf() and scanf() methods (CO6) (BT 3)

- a) fscanf() can read from standard input whereas scanf() specifies a stream from which to read
- b) fscanf() can specifies a stream from which to read whereas scanf() can read only from standard input
- c) fscanf() and scanf() has no difference in their functions
- d) fscanf() and scanf() can read from specified stream

Answer:b) fscanf() can specifies a stream from which to read whereas scanf() can read only from standard input

4. What is the value of EOF which is defined in stdio.h header file?(CO6) (BT 1)

- a) 1
- b) 0
- c) NULL

d) – 1

Answer: d) – 1

5. Infer the statement: "fwrite() can be used only with files that are opened in binary mode".(CO6) (BT 2)

a) true

b) false

Answer: a) true

6. Recall the function of fputs()?(CO6)(BT1)

a) read a line from a file

b) read a character from a file

c) write a character to a file

d) write a line to a file

Answer:d) write a line to a file

7. Summarize the action done by the following C code snippet.(CO6)(BT1)

```
char *gets(char *s)
```

a) reads the next input line into the array s

b) writes the line into the array s

c) reads the next input character into the array s

d) write a character into the array

Answer:a) reads the next input line into the array s

8. Tell the function which will return the current file position for stream?(CO6)(BT1)

a) fgetpos()

b) fseek()

c) ftell()

d) fsetpos()

Answer:c) ftell()

9. Before we can read or write to a file in C, what do we need to do? (CO6)(BT 1)

a) open the file

b) close the file

c) print the content of the file

d) NONE

Answer:a) open the file

10. Which flag is used in fopen function to append data to an existing file instead of recreating the file?(CO6)(BT 1)

a) a

b) r

c) w

d) W+

Answer:a) a

11. If the mode includes b after the initial letter in fopen function then what does it indicates?(CO6)(BT 1)

- a) text file
- b) big text file
- c) binary file
- d) empty text

Answer: c) binary file

12. Select the correct statement about FILE \*fp. (CO6)(BT 3)

- a) FILE is a keyword in C for representing files and fp is a variable of FILE type.
- b) FILE is a stream
- c) FILE is a buffered stream
- d) FILE is a structure and fp is a pointer to the structure of FILE type

Answer: d) FILE is a structure and fp is a pointer to the structure of FILE type

13. Summarize the first and second arguments of fopen().(CO6)(BT 2)

- a) name of the file,mode
- b) name of the user,mode
- c) file pointer , mode
- d) None

Answer:a) a) name of the file,mode

14. Find the type of FILE. (CO6) (BT1)

- a) int type
- b) char \* type
- c) struct type
- d) float

Answer:c) struct type

15. Identify the main reason to prefer fseek() over rewind().(CO6) (BT3)

- a) rewind() doesn't work for empty files
- b) rewind() may fail for large files
- c) In rewind, there is no way to check if the operations completed successfully
- d) All of the above

Answer:c) In rewind, there is no way to check if the operations completed successfully

16. For binary files, which of the following must be appended to the mode string. (CO6) (BT1)

- a) b
  - b) B
  - c) binary
  - d) 01
- Answer: a) b

17. Select the reason for closing a file in C language.(CO6)(BT3)
- a) fclose(fp) closes a file to release the memory used in opening a file.
  - b) Closing a file clears Buffer contents from RAM or memory.
  - c) Unclosed files occupy memory and PC hangs when on low memory.
  - d) All of these
- Answer: d) All of these

18. Choose the correct C functions to read or write to a Text file. (CO6)(BT1)
- a) fprintf(), fscanf()
  - b) fread(), fwrite()
  - c) fprintf(), scanf()
  - d) read(), write()
- Answer:a) fprintf(), fscanf()

19. Choose the correct C functions to read or write to a binary file. (CO6)(BT1)
- a) fprintf(), fscanf()
  - b) fread(), fwrite()
  - c) fprintf(), scanf()
  - d) read(), write()
- Answer:b) fread(), fwrite()

20. Which C function is used to move current pointer to the beginning of file.?(CO6)(BT1)
- a) revise(fp)
  - b) rewind(fp)
  - c) ftell(fp)
  - d) frewind(fp)
- Answer:b) rewind(fp)

## B. SHORT AND LONG QUESTIONS

1. Compare and contrast Text file and binary file.(CO6)(BT3)
2. Explain different modes to open a text file. (CO6)(BT2)
3. How does file pointer created in C program? (CO6)(BT1)
4. Discuss in detail how files are accessed randomly in C with suitable example. (CO6)(BT3)
5. Why do we need a file? (CO6)(BT1)
6. Outline the functions which are used to read data from a text file. (CO6)(BT2)
7. Develop a C program to copy the content of one file to another file. (CO6)(BT3)

8. Show how to write data to a binary file. (CO6)(BT2)
9. Infer the statement: "Reading from a text file and reading from a binary file can be done using same function X". Are you agreeing with this statement? Summarize your opinion. Outline the functions which are used to read data from a text file. (CO6)(BT2)
10. Develop a C program to read from text file which contains student name, roll no and marks of 3 subjects. Calculate and Display the percentage of marks for each student. (CO6)(BT3)

**Questions:**

1. What are preprocessor commands? Give examples.
2. Difference between Macro and Function.
3. Distinguish between function and preprocessor directive.
4. Difference between #ifdef and #ifndef with example .
5. Give a brief note about Error directive.

**MCQ**

1. What will be the output of the program?

```
#include<stdio.h>

#define int char

void main()
{
    int i = 65;
    printf("sizeof(i)=%d", sizeof(i));
}
```

- A. Sizeof(i)=2;
- B. Sizeof(i)=1;
- C. Compilation error
- D. None

Answer:B

2. What will be the output of the program?

```
#include<stdio.h>
#define a 10
void main()
{
    #define a 50
```

```
    printf("%d", a);
}
A. 50
B. 10
C. Compilation error
D. None
```

Answer: A

3. What will be the output of the program?

```
#include <stdio.h>
#define p 17;
int main()
{
    printf("%d", p);
    return 0;
}
```

- A. 17
- B. Run-time error
- C. Compilation error
- D. Garbage value

Answer: A

4. In which stage the following code

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

gets replaced by the contents of the file *stdio.h*

- A. During editing
- B. During linking
- C. During Preprocessor
- D. During execution

Answer: C

5. What will be the output of the program?

```
#include<stdio.h>
#define SQR(x) (x*x)
int main()
{
    int a, b=3;
    a = SQR(b+2);
    printf("%d\n", a);
    return 0;
}
```

- A. 25
- B. 11
- C. Error
- D. Garbage value

Answer: B



# CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)

Kokapet(Village), Gandipet, Hyderabad, Telangana-500075. [www.cbit.ac.in](http://www.cbit.ac.in)

Approved by  
AICTE  
Regd. No. 2566  
1979

NAAC  
Accredited  
A+  
Program

Accredited by  
NAAC  
Grade - A+

Wife of  
NIRF  
RANKED  
100 IN  
TECHNICAL  
COLLEGES

ISO  
9001:2015

COMMITTED TO  
RESEARCH,  
INNOVATION AND  
EDUCATION

41  
years

Course Code: 20CS C01

Course Title: PROGRAMMING FOR PROBLEM SOLVING

| Module/<br>Unit | Assignment<br>Question NO | Theme/ Assignment Question/Case Study                                                                                                                                                                                                          | CO No   |
|-----------------|---------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
| Unit-1          | 1.1                       | <p>Write a C program to print your name, date of birth. and mobile number. <u><a href="#">Go to the editor</a></u></p> <p><i>Expected Output:</i></p> <p>Name : Alexandra Abramov</p> <p>DOB : July 14, 1975</p> <p>Mobile : 99-9999999999</p> | CO3     |
|                 | 1.2                       | Write a C program that accepts two integers from the user and calculate the sum of the two integers.                                                                                                                                           | CO1,CO3 |
|                 | 1.3                       | Write a C program to convert specified days into years, weeks and days                                                                                                                                                                         | CO2,CO3 |
|                 | 1.4                       | Write a C program to take a list of n elements from the user. Store it in an array. Reverse the list.                                                                                                                                          | CO5     |
|                 | 1.5                       | Write a C program that accepts an employee's ID, total worked hours of a month and the amount he received per hour. Print the employee's ID and salary (with two decimal places) of a particular                                               | CO2,CO5 |

|        |                                                                                                                                                                                                                                                      |                                                                                                                                                                               |             |
|--------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
|        |                                                                                                                                                                                                                                                      | month.                                                                                                                                                                        |             |
| 1.6    | Determine the size, minimum and maximum value following data types. Please specify if your machine is 32 bit or 64 bits in the answer.<br>• char • unsigned char • short • int • unsigned int • unsigned long • float<br>Hint: Use sizeof() operator | CO1,CO3                                                                                                                                                                       |             |
| 1.7    | Demonstrate the working of type conversion techniques.                                                                                                                                                                                               | CO3                                                                                                                                                                           |             |
| 1.8    | Converting between Celsius and Fahrenheit (ConvertTemperature.c)<br>* Celsius = $(5/9)(Fahrenheit - 32)$<br>* Fahrenheit = $(9/5)Celsius + 32$                                                                                                       | CO1,CO4                                                                                                                                                                       |             |
| 1.9    | Demonstrate the working of increment (++) and decrement (--) Operator                                                                                                                                                                                | CO2                                                                                                                                                                           |             |
| 1.10   | Write a C program to calculate a bike's average consumption from the given total distance (integer value) traveled (in km) and spent fuel (in liters, float number – 2 decimal point).                                                               | CO2                                                                                                                                                                           |             |
| Unit-2 | 2.1                                                                                                                                                                                                                                                  | Write logical expressions that tests whether a given character variable c is<br>• lower case letter • upper case letter • digit • white space (includes space, tab, new line) | CO3         |
|        | 2.2                                                                                                                                                                                                                                                  | Write a C program that calculates the HCF and LCM of two numbers using function.                                                                                              | CO1,CO2,CO4 |
|        | 2.3                                                                                                                                                                                                                                                  | List non-prime from 1 to an upperbound                                                                                                                                        | CO1,CO3     |
|        | 2.4                                                                                                                                                                                                                                                  | Write a C function for the following problem: Given a positive integer n, print the binary representation of n.                                                               | CO1,CO3,CO2 |
|        | 2.5                                                                                                                                                                                                                                                  | Write a Program to check whether a number is amicable number                                                                                                                  | CO1,CO3     |
|        | 2.6                                                                                                                                                                                                                                                  | Write a program to print the triangular number pattern.                                                                                                                       | CO1,CO3     |
|        | 2.7                                                                                                                                                                                                                                                  | Write a C program to display and find the sum of the series 1+11+111+. 111 upto n. For eg. If n=4, the series is : 1+11+111+1111. Take the value of 'n' as                    | CO1,CO3,CO2 |

|                 |     | input from the user.                                                                                                                                                                                                                                                                                                                                                                                                                                        |             |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
|-----------------|-----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|-----|------|----------|-----|-----|-------------|-----|-----|--------------|-----|-----|-----------------|-----|-----|---------|
| 2.8             |     | Program to calculate Electricity Bill                                                                                                                                                                                                                                                                                                                                                                                                                       | CO2,CO3     |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
| 2.9             |     | <p>Given a number n, the task is to print the triangular patterns of alphabets of the length n. First print the n characters then decrement one from the beginning in each line.</p>                                                                                                                                                                                                                                                                        | CO1,CO3     |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
| 2.10            |     | <p>Accept the salary of an employee from the user. Calculate the gross salary on the following basis:</p> <table> <thead> <tr> <th>Basic</th> <th>HRA</th> <th>DA .</th> </tr> </thead> <tbody> <tr> <td>1 - 4000</td> <td>10%</td> <td>50%</td> </tr> <tr> <td>4001 - 8000</td> <td>20%</td> <td>60%</td> </tr> <tr> <td>8001 - 12000</td> <td>25%</td> <td>70%</td> </tr> <tr> <td>12000 and above</td> <td>30%</td> <td>80%</td> </tr> </tbody> </table> | Basic       | HRA | DA . | 1 - 4000 | 10% | 50% | 4001 - 8000 | 20% | 60% | 8001 - 12000 | 25% | 70% | 12000 and above | 30% | 80% | CO2,CO3 |
| Basic           | HRA | DA .                                                                                                                                                                                                                                                                                                                                                                                                                                                        |             |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
| 1 - 4000        | 10% | 50%                                                                                                                                                                                                                                                                                                                                                                                                                                                         |             |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
| 4001 - 8000     | 20% | 60%                                                                                                                                                                                                                                                                                                                                                                                                                                                         |             |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
| 8001 - 12000    | 25% | 70%                                                                                                                                                                                                                                                                                                                                                                                                                                                         |             |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
| 12000 and above | 30% | 80%                                                                                                                                                                                                                                                                                                                                                                                                                                                         |             |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
| 2.11            |     | Write a C program to read an amount (integer value) and break the amount into smallest possible number of bank notes using function.                                                                                                                                                                                                                                                                                                                        | CO1,CO3,CO4 |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
| Unit-3          | 3.1 | Find the mean and standard deviation of numbers kept in an array                                                                                                                                                                                                                                                                                                                                                                                            | CO5         |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
|                 | 3.2 | Sum the odd and even numbers, respectively, from 1 to a given upperbound.<br>* Also compute the absolute difference.                                                                                                                                                                                                                                                                                                                                        | CO4,CO5     |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
|                 | 3.3 | C Program to Check Two Matrices are Equal or Not                                                                                                                                                                                                                                                                                                                                                                                                            | CO5         |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
|                 | 3.4 | The function checks if the array elements are consecutive<br>If elements are consecutive, then returns true, else returns false                                                                                                                                                                                                                                                                                                                             | CO5         |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |
|                 | 3.5 | Find out the ugly prime number<br>Desc: The given number is ugly prime                                                                                                                                                                                                                                                                                                                                                                                      | CO1,CO5     |     |      |          |     |     |             |     |     |              |     |     |                 |     |     |         |

|        |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |             |
|--------|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------------|
|        |     | <p>number if it's prime factor contains only among 2,3 or 5.<br/>e.g. <math>20=2*2*5</math> is ugly prime number<br/> <math>14=2*7</math> is not a ugly prime number<br/> So write a C function which takes values from 1 to n and returns the number of ugly primes number in it.<br/> input: 20<br/> output: 3</p>                                                                                                                                                                                                                                                                                                                                                                              |             |
| 3.6    |     | Write a C program that reads an expression and evaluates it.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | CO1,CO2     |
| 3.7    |     | 8. Write a C program to input n numbers in an array, calculate the sum of all even numbers and all odd numbers in the array and print the larger sum. Example: If the array contains the following elements: 2, 3, 3, 5, 4, 8, 7, 11, 2 The sum of all even elements is $2+4+8+2=16$ Sum of all odd elements is $3+3+5+7+11=29$ Therefore, the output should be 29.                                                                                                                                                                                                                                                                                                                               | CO2,CO5     |
| 3.8    |     | Sum of all numerical values (positive integers) embedded in a sentence                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | CO2,CO3,CO5 |
| 3.9    |     | Find the element(s) which occurs most frequently in a given sequence                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | CO2,CO5     |
| 3.10   |     | <p>Write a C program to calculate the volume of the following shapes: Cube, Cuboid, Sphere, Cylinder and Cone. Ask the user which one s/he wants to calculate, and take the appropriate required inputs. Then print the result. The input should be taken in the main function and calculations for every solid should be done in a separate function by passing appropriate arguments.</p> <p>Example: If the user chooses the option for cube, only one input is required i.e., the side. The volume is then calculated and printed. If the user chooses the option for cuboid, only three inputs are required i.e., length, breadth and height. The volume is then calculated and printed.</p> | CO1,CO5     |
| Unit-4 | 4.1 | Write a program in C to demonstrate the use of &(address of) and *(value at address) operator.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | CO5         |

|        |      |                                                                                                                                                         |         |
|--------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------|---------|
|        | 4.2  | Input size and values in two arrays A and B, of max size 100. Then, compute A[i]+B[i] and store in array C. And compute A[i]*B[i] and store in array D. | CO5     |
|        | 4.3  | Write a program in C to print all permutations of a given string using pointers.                                                                        | CO1,CO5 |
|        | 4.4  | Write a program in C to find the largest element using Dynamic Memory Allocation                                                                        | CO5     |
|        | 4.5  | Write a program in C to Calculate the length of the string using a pointer.                                                                             | CO5     |
|        | 4.6  | Write a program in C to find the factorial of a given number using pointers                                                                             | CO5     |
|        | 4.7  | Write a program in C to count the number of vowels and consonants in a string using a pointer.                                                          | CO5     |
|        | 4.8  | Write a program in C to sort an array using Pointer.                                                                                                    | CO5     |
|        | 4.9  | Write a program in C to compute the sum of all elements in an array using pointers                                                                      | CO5     |
|        | 4.10 | Write a program in C to print the elements of an array in reverse order.                                                                                | CO5     |
| Unit-5 | 5.1  | Write a program in C to write multiple lines in a text file                                                                                             | CO6     |
|        | 5.2  | Write a program in C to Find the Number of Lines in a Text File                                                                                         | CO6     |
|        | 5.3  | Write a program in C to find the content of the file and number of lines in a Text File                                                                 | CO6     |
|        | 5.4  | Write a program in C to count a number of words and characters in a file.                                                                               | CO6     |
|        | 5.5  | Write a program in C to delete a specific line from a file.                                                                                             | CO6     |
|        | 5.6  | Write a program in C to replace a specific                                                                                                              | CO6     |

|      |  |                                                                          |     |
|------|--|--------------------------------------------------------------------------|-----|
|      |  | line with another text in a file                                         |     |
| 5.7  |  | Write a program in C to append multiple lines at the end of a text file. | CO6 |
| 5.8  |  | Write a program in C to copy a file in another name                      | CO6 |
| 5.9  |  | Write a program in C to merge two files and write it in a new file.      | CO6 |
| 5.10 |  | Remove a file from the disk                                              | CO6 |

|            |             |
|------------|-------------|
| Assignment | Unit-1      |
| Assignment | 1.1 to 1.10 |
| Date       |             |

Solution/Answer:

1.1

```
#include <stdio.h>
int main()
{
    printf("Name : Alexandra Abramov\n");
    printf("DOB : July 14, 1975\n");
    printf("Mobile : 99-9999999999\n");
    return(0);
}
```

O/P:

```
Name : Alexandra Abramov
DOB : July 14, 1975
Mobile : 99-9999999999
```

1.2

```
#include <stdio.h>
int main()
```

```
{  
    int x, y, sum;  
    printf("\nInput the first integer: ");  
    scanf("%d", &x);  
    printf("\nInput the second integer: ");  
    scanf("%d", &y);  
    sum = x + y;  
    printf("\nSum of the above two integers = %d\n", sum);  
    return 0;  
}  
O/P: Input the first integer: 25
```

Input the second integer: 38

Sum of the above two integers = 63

1.3

```
#include <stdio.h>  
int main()  
{  
    int days, years, weeks;  
  
    days = 1329;  
  
    // Converts days to years, weeks and days  
    years = days/365;  
    weeks = (days % 365)/7;  
    days = days- ((years*365) + (weeks*7));  
  
    printf("Years: %d\n", years);  
    printf("Weeks: %d\n", weeks);  
    printf("Days: %d \n", days);  
  
    return 0;  
}
```

O/P:

Years: 3

Weeks: 33

Days: 3

1.4

```
#include <stdio.h>  
int main()  
{  
    char char1 = 'X';  
    char char2 = 'M';
```

```
char char3 = 'L';
printf("The reverse of %c%c%c is %c%c%c\n",
       char1, char2, char3,
       char3, char2, char1);

return(0);
}
```

O/P:

The reverse of XML is LMX

1.5

```
#include <stdio.h>
int main() {
char id[10];
int hour;
double value, salary;
printf("Input the Employees ID(Max. 10 chars): ");
scanf("%s", &id);
printf("\nInput the working hrs: ");
scanf("%d", &hour);
printf("\nSalary amount/hr: ");
scanf("%lf", &value);
salary = value * hour;
printf("\nEmployees ID = %s\nSalary = U$ %.2lf\n", id,salary);
return 0;
}
```

O/P:

Input the Employees ID(Max. 10 chars): 0342

Input the working hrs: 8

Salary amount/hr: 15000

Employees ID = 0342

Salary = U\$ 120000.00

1.6

```
#include <stdio.h>
int main() {
printf("sizeof(char) is %d bytes.\n", sizeof(char));
printf("sizeof(short) is %d bytes.\n", sizeof(short));
printf("sizeof(int) is %d bytes.\n", sizeof(int));
printf("sizeof(long) is %d bytes.\n", sizeof(long));
printf("sizeof(long long) is %d bytes.\n", sizeof(long long));
printf("sizeof(float) is %d bytes.\n", sizeof(float));
printf("sizeof(double) is %d bytes.\n", sizeof(double));
printf("sizeof(long double) is %d bytes.\n", sizeof(long double));
return 0;
}
```

O/P:

```
sizeof(char) is 1 bytes.
sizeof(short) is 2 bytes.
sizeof(int) is 4 bytes.
sizeof(long) is 4 bytes.
sizeof(long long) is 8 bytes.
sizeof(float) is 4 bytes.
sizeof(double) is 8 bytes.
sizeof(long double) is 12 bytes.
```

1.7

```
int anInt = 12345;
float aFloat = 55.6677;
double aDouble = 11.2233;
char aChar = 'a';
char aStr[] = "Hello";
printf("The int is %d.\n", anInt);
//The int is 12345.
printf("The float is %f.\n", aFloat);
//The float is 55.667702.
printf("The double is %lf.\n", aDouble);
//The double is 11.223300.
printf("The char is %c.\n", aChar);
//The char is a.
printf("The string is %s.\n", aStr);
//The string is Hello.
printf("The int (in hex) is %x.\n", anInt);
//The int (in hex) is 3039.
printf("The double (in scientific) is %le.\n", aDouble);
//The double (in scientific) is 1.122330e+01.
printf("The float (in scientific) is %E.\n", aFloat);
//The float (in scientific) is 5.566770E+01.
```

1.8

```
#include <stdio.h>
int main() {
    double celsius, fahrenheit;

    printf("Enter the temperature in celsius: ");
    scanf("%lf", &celsius);
    fahrenheit = celsius * 9 / 5 + 32;
    // 9/5*celsius + 32 gives wrong answer! Why?
    printf("%.2lf degree C is %.2lf degree F\n", celsius, fahrenheit);

    printf("Enter the temperature in fahrenheit: ");
    scanf("%lf", &fahrenheit);
```

```

celsius = (fahrenheit - 32) * 5 / 9;
// 5/9*(fahrenheit - 32) gives wrong answer! Why?
printf("%.2lf degree F is %.2lf degree C\n", fahrenheit, celsius);
return 0;
}

1.9
#include <stdio.h>
int main()
{
    int mark = 76;      // declare & assign
    printf("%d\n", mark); // 76
    mark++;           // increase by 1 (post-increment)
    printf("%d\n", mark); // 77
    ++mark;           // increase by 1 (pre-increment)
    printf("%d\n", mark); // 78
    mark = mark + 1;   // also increase by 1 (or mark += 1)
    printf("%d\n", mark); // 79
    mark--;            // decrease by 1 (post-decrement)
    printf("%d\n", mark); // 78

    --mark;            // decrease by 1 (pre-decrement)
    printf("%d\n", mark); // 77
    mark = mark - 1;   // also decrease by 1 (or mark -= 1)
    printf("%d\n", mark); // 76
    return 0;
}

1.10
#include <stdio.h>
int main()
{
    int x;
    float y;
    printf("Input total distance in km: ");
    scanf("%d", &x);
    printf("Input total fuel spent in liters: ");
    scanf("%f", &y);
    printf("Average consumption (km/lt) %.3f ", x/y);
    printf("\n");
    return 0;
}
O/P: Input total distance in km: 350
Input total fuel spent in liters: 5
Average consumption (km/lt) 70.000

```

|                   |             |
|-------------------|-------------|
| Assignment        | Unit-2      |
| Assignment Number | 2.1 to 2.11 |
| Date              |             |

Solution/Answer:

2.1

```
#include <stdio.h>
int main()
{
    char ch;

    /* Input character from user */
    printf("Enter any character: ");
    scanf("%c", &ch);

    if(ch >= 'A' && ch <= 'Z')
    {
        printf("'"%c' is uppercase alphabet.", ch);
    }
    else if(ch >= 'a' && ch <= 'z')
    {
        printf("'"%c' is lowercase alphabet.", ch);
    }
    else
    {
        printf("'"%c' is not an alphabet.", ch);
    }

    return 0;
}
```

2.2

```
#include <stdio.h>
long gcd(long, long);

int main() {
    long x, y, hcf, lcm;
    printf("Enter two integers\n");
    scanf("%ld%ld", &x, &y);
    hcf = gcd(x, y);
    lcm = (x*y)/hcf;
    printf("Greatest common divisor of %ld and %ld = %ld\n", x, y, hcf);
    printf("Least common multiple of %ld and %ld = %ld\n", x, y, lcm);
    return 0;
}
```

```

long gcd(long x, long y) {
    if (x == 0) {
        return y;
    }
    while (y != 0) {
        if (x > y)
            x = x - y;
        else
            y = y - x;
    }
    return x;
}

```

O/P:

```

Enter two integers
9
24
Greatest common divisor of 9 and 24 = 3
Least common multiple of 9 and 24 = 72

```

2.3

```

#include <stdio.h>
#include <math.h>
int main() {
    int upperbound, number, maxFactor, factor;
    printf("Enter the upperbound: ");
    scanf("%d", &upperbound);
    for (number = 2; number <= upperbound; ++number) {
        // Not a prime, if there is a factor between 2 and sqrt(number)
        maxFactor = (int)sqrt(number);
        for (factor = 2; factor <= maxFactor; ++factor) {
            if (number % factor == 0) { // Factor?
                printf("%d ", number);
                break; // A factor found, no need to search for more factors
            }
        }
    }
    printf("\n");
    return 0;
}

```

O/P:

```

Enter the upperbound: 10
4 6 8 10

```

2.4

```

#include <stdio.h>
#include <math.h>
long decimalToBinary(int decimalnum)

```

```

{
    long binarynum = 0;
    int rem, temp = 1;
    while (decimalnum!=0)
    {
        rem = decimalnum%2;
        decimalnum = decimalnum / 2;
        binarynum = binarynum + rem*temp;
        temp = temp * 10;
    }
    return binarynum;
}

int main()
{
    int decimalnum;
    printf("Enter a Decimal Number: ");
    scanf("%d", &decimalnum);
    printf("Equivalent Binary Number is: %ld", decimalToBinary(decimalnum));
    return 0;
}

2.5
int divSum(int n)
{
    // Sum of divisors
    int result = 0;

    // find all divisors which divides 'num'
    for (int i = 2; i <= sqrt(n); i++)
    {
        // if 'i' is divisor of 'n'
        if (n % i == 0)
        {
            // if both divisors are same
            // then add it once else add
            // both
            if (i == (n / i))
                result += i;
            else
                result += (i + n/i);
        }
    }

    return (result + 1);
}

```

```

// Returns true if x and y are Amicable
// else false.
bool areAmicable(int x, int y)
{
    if (divSum(x) != y)
        return false;

    return (divSum(y) == x);
}
int main() {
    int x = 220, y = 284;
    if (areAmicable(x, y))
        printf("Yes");
    else
        printf("No");
    return 0;
}
O/P: Yes

```

2.6

```

#include <stdio.h>
// Function to find triangular number
void triangular_series(int n)
{
    int i, j = 1, k = 1;

    // For each iteration increase j by 1
    // and add it into k
    for (i = 1; i <= n; i++) {
        printf("%d ", k);
        j = j + 1; // Increasing j by 1
        k = k + j; // Add value of j into k and update k
    }
}
// Driven Function
int main()
{
    int n = 5;
    triangular_series(n);
    return 0;
}
O/P: 1 3 6 10 15....

```

2.7

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

```

void main()
{
    int n,i;
    long sum=0;
    long int t=1;
    printf("Input the number of terms : ");
    scanf("%d",&n);
    for(i=1;i<=n;i++)
    {
        printf("%ld ",t);
        if (i<n)
        {
            printf("+ ");
        }
        sum=sum+t;
        t=(t*10)+1;
    }
    printf("\nThe Sum is : %ld\n",sum);
}

```

O/P:

Input the number of terms : 5  
 1 + 11 + 111 + 1111 + 11111  
 The Sum is : 12345

2.8

```

int calculateBill(int units)
{
    if (units <= 100)
    {
        return units * 10;
    }
    else if (units <= 200)
    {
        return (100 * 10) +
               (units - 100) * 15;
    }
    else if (units <= 300)
    {
        return (100 * 10) +
               (100 * 15) +
               (units - 200) * 20;
    }
    else if (units > 300)
    {
        return (100 * 10) +

```

```

        (100 * 15) +
        (100 * 20) +
        (units - 300) * 25;
    }
    return 0;
}

// Driver Code
int main()
{
    int units = 250;
    printf(calculateBill(units));
}

```

O/P: 3500

2.9

```

#include <stdio.h>
int pattern( int n){
    int i, j;
    for (i = 1; i <= n; i++) {
        for (j = i; j <= n; j++) {
            printf("%c", 'A' - 1 + j);
        }
        printf("\n");
    }
    return 0;
}
int main(){
    int n = 5;
    pattern(n);
    return 0;
}

```

O/P:

```

ABCDE
BCDE
CDE
DE
E

```

2.10

```

#include <stdio.h>
int main()
{
    float basic, gross, da, hra;

```

```

/* Input basic salary of employee */
printf("Enter basic salary of an employee: ");
scanf("%f", &basic);
/* Calculate D.A and H.R.A according to specified conditions */
if(basic <= 10000)
{
    da = basic * 0.8;
    hra = basic * 0.2;
}
else if(basic <= 20000)
{
    da = basic * 0.9;
    hra = basic * 0.25;
}
else
{
    da = basic * 0.95;
    hra = basic * 0.3;
}

/* Calculate gross salary */
gross = basic + hra + da;
printf("GROSS SALARY OF EMPLOYEE = %.2f", gross);
return 0;
}
2.11
#include <stdio.h>
void calculate(int amt)
{
total = (int)amt/100;

printf("There are: ");

printf("\n%d Note(s) of 100.00\n", total);

amt = amt-(total*100);

total = (int)amt/50;

printf("%d Note(s) of 50.00\n", total);

amt = amt-(total*50);

total = (int)amt/20;

printf("%d Note(s) of 20.00\n", total);

```

```

amt = amt-(total*20);

total = (int)amt/10;

printf("%d Note(s) of 10.00\n", total);

amt = amt-(total*10);

total = (int)amt/5;

printf("%d Note(s) of 5.00\n", total);

amt = amt-(total*5);

total = (int)amt/2;

printf("%d Note(s) of 2.00\n", total);

amt = amt-(total*2);

total = (int)amt/1;

printf("%d Note(s) of 1.00\n", total);

return 0;
}

```

```

int main() {
    int amt, total;
    printf("Input the amount: ");
    scanf("%d",&amt);
    calculate(375)
}

```

o/p: Input the amount: 375

There are:

3 Note(s) of 100.00  
 1 Note(s) of 50.00  
 1 Note(s) of 20.00  
 0 Note(s) of 10.00  
 1 Note(s) of 5.00  
 0 Note(s) of 2.00  
 0 Note(s) of 1.00

|            |  |
|------------|--|
| Assignment |  |
| Assignment |  |

|      |  |
|------|--|
| Date |  |
|------|--|

 Solution/Answer: |

3.1

```
#include <stdio.h>
#include <math.h>
#define SIZE 7
int main() {
    int marks[] = { 74, 43, 58, 60, 90, 64, 70};
    int sum = 0;
    int sumSq = 0;
    double mean, stdDev;
    int i;
    for (i = 0; i < SIZE; ++i) {
        sum += marks[i];
        sumSq += marks[i] * marks[i];
    }
    mean = (double)sum/SIZE;
    printf("Mean is %.2lf\n", mean);
    stdDev = sqrt((double)sumSq/SIZE - mean*mean);
    printf("Std dev is %.2lf\n", stdDev);
    return 0;
}
```

3.2

```
#include<stdio.h>
int main()
{
    int Size, i, a[10];
    int Even_Sum = 0, Odd_Sum = 0;
    printf("\n Please Enter the Size of an Array : ");
    scanf("%d", &Size);
    printf("\nPlease Enter the Array Elements\n");
    for(i = 0; i < Size; i++)
    {
        scanf("%d", &a[i]);
    }
    for(i = 0; i < Size; i++)
    {
        if(a[i] % 2 == 0)
        {
            Even_Sum = Even_Sum + a[i];
        }
        else
        {
            Odd_Sum = Odd_Sum + a[i];
        }
    }
}
```

```

    }
}

printf("\n The Sum of Even Numbers in this Array = %d ", Even_Sum);
printf("\n The Sum of Odd Numbers in this Array = %d ", Odd_Sum);
return 0;
}

```

O/P:

```

Please Enter the Size of an Array : 5
Please Enter the Array Elements
15 25 35 10 90
The Sum of Even Numbers in this Array = 100
The Sum of Odd Numbers in this Array = 75

```

3.3

```

#include<stdio.h>
int main()
{
    int i, j, rows, columns, a[10][10], b[10][10], isEqual;
    printf("\n Please Enter Number of rows and columns : ");
    scanf("%d %d", &i, &j);
    printf("\n Please Enter the First Matrix Elements\n");
    for(rows = 0; rows < i; rows++)
    {
        for(columns = 0; columns < j; columns++)
        {
            scanf("%d", &a[rows][columns]);
        }
    }
    printf("\n Please Enter the Second Matrix Elements\n");
    for(rows = 0; rows < i; rows++)
    {
        for(columns = 0; columns < j; columns++)
        {
            scanf("%d", &b[rows][columns]);
        }
    }
    isEqual = 1;

    for(rows = 0; rows < i; rows++)
    {
        for(columns = 0; columns < j; columns++)
        {
            if(a[rows][columns] != b[rows][columns])
            {
                isEqual = 0;
                break;
            }
        }
    }
}

```

```

    }
}
}
if(isEqual == 1)
{
printf("\n Matrix a is Equal to Matrix b");
}
else
{
printf("\n Matrix a is Not Equal to Matrix b");
}
return 0;
}

```

O/P:

```

Please Enter Number of rows and columns :  2 2
Please Enter the First Matrix Elements
1 2
3 4
Please Enter the Second Matrix Elements
1 2
3 4
Matrix a is Equal to Matrix b

```

### 3.4

Consecutive numbers

```

bool areConsecutive(int arr[], int n)
{
    if ( n < 1 )
        return false;

    /* 1) Get the minimum element in array */
    int min = getMin(arr, n);

    /* 2) Get the maximum element in array */
    int max = getMax(arr, n);

    /* 3) max - min + 1 is equal to n then only check all elements */
    if (max - min + 1 == n)
    {
        int i;
        for(i = 0; i < n; i++)
        {
            int j;

            if (arr[i] < 0)
                j = -arr[i] - min;
        }
    }
}

```

```

        else
            j = arr[i] - min;

        // if the value at index j is negative then
        // there is repetition
        if (arr[j] > 0)
            arr[j] = -arr[j];
        else
            return false;
    }

    /* If we do not see a negative value then all elements
       are distinct */
    return true;
}

return false; // if (max - min + 1 != n)
}

/* UTILITY FUNCTIONS */
int getMin(int arr[], int n)
{
    int min = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] < min)
            min = arr[i];
    return min;
}

int getMax(int arr[], int n)
{
    int max = arr[0];
    for (int i = 1; i < n; i++)
        if (arr[i] > max)
            max = arr[i];
    return max;
}

/* Driver program to test above functions */
int main()
{
    int arr[] = {1, 4, 5, 3, 2, 6};
    int n = sizeof(arr)/sizeof(arr[0]);
    if(areConsecutive(arr, n) == true)
        printf(" Array elements are consecutive ");
    else

```

```
    printf(" Array elements are not consecutive ");
    getchar();
    return 0;
}
```

notes

### 3.5

Read the values from a given array.

```
int digProduct(int n)
{
    int product = 1;

    while (n != 0) {
        product = product * (n % 10);
        n = n / 10;
    }

    return product;
}
```

// Function that returns true if n

// is prime else returns false

```
bool isPrime(int n)
{
    // Corner cases
    if (n <= 1)
        return false;
    if (n <= 3)
        return true;

    // This is checked so that we can skip
    // middle five numbers in below loop
    if (n % 2 == 0 || n % 3 == 0)
        return false;
```

```
    for (int i = 5; i * i <= n; i = i + 6)
        if (n % i == 0 || n % (i + 2) == 0)
            return false;
```

```
    return true;
}
```

// Function to return the smallest

// prime number greater than N

```
int nextPrime(int N)
```

```

{
    // Base case
    if (N <= 1)
        return 2;

    int prime = N;
    bool found = false;

    // Loop continuously until isPrime returns
    // true for a number greater than n
    while (!found) {
        prime++;

        if (isPrime(prime))
            found = true;
    }

    return prime;
}

// Function to check Pointer-Prime numbers
bool isPointerPrime(int n)
{
    if (isPrime(n)
        && (n + digProduct(n) == nextPrime(n)))
        return true;
    else
        return false;
}

// Driver Code
int main()
{
    // Given Number N
    int N = 23;

    // Function Call
    if (isPointerPrime(N))
        cout << "Yes";
    else
        cout << "No";
    return 0;
}
Output:
Yes

```

3.6  
#include <stdio.h>  
#include<string.h>

int addsub();  
int muldiv();  
int term();

char input[101];  
int pos = 0;

int term(){  
 int n = 0;

if(input[pos] == '('){  
 pos++;  
 n = addsub();  
  
 if(input[pos] == ')'){  
 pos++;  
 return n;  
 }  
 }else{  
 while('0' <= input[pos] && input[pos] <= '9'){  
 n = n\*10 + (input[pos] - '0');  
 pos++;  
 }  
 }  
 return n;  
}

int muldiv(){  
 int first,second;  
  
 first = term();  
 for(;;){  
 if(input[pos] == '\*'){  
 pos++;  
 second = term();  
 first \*= second;  
 }else if(input[pos] == '/')  
 {  
 pos++;  
 second = term();  
 first /= second;  
 }else{

```

        return first;
    }
}
}

int addsub(){
    int first,second;

    first = muldiv();

    for(;;){
        if(input[pos] == '+'){
            pos++;
            second = muldiv();
            first += second;
        }else if(input[pos] == '-'){
            pos++;
            second = muldiv();
            first -= second;
        }else{
            return first;
        }
    }
}

int main(){
    int n,i,j;
    printf("Input an expression using +, -, *, / operators:\n");
    scanf("%s",input);
    printf("%d\n",addsub());
    return 0;
}

```

O/P:

Input an expression using +, -, \*, / operators:

1+6\*8-4/2

47

Input an expression using +, -, \*, / operators:

25/5-6\*7+2

-35

3.7

#include <stdio.h>

int main()

{

int cell\_data[11][11];

int i, j, n, sum\_val;

printf("Input number of rows/columns:\n");

```

scanf("%d", &n);
printf("Input the cell value\n");
if(n>0)
{
for(i=0;i<n;i++){
    printf("\nRow %d input cell values\n",i);
    for(j=0;j<n;j++){
        scanf("%d", &cell_data[i][j]);
    }
}
printf("\nResult:\n");
for(i=0;i<n;i++){
    sum_val=0;
    for(j=0;j<n;j++){
        sum_val+=cell_data[j][i];
    }
    cell_data[n][i]=sum_val;
}

for(i=0;i<n;i++){
    sum_val=0;
    for(j=0;j<n;j++){
        sum_val+=cell_data[i][j];
    }
    cell_data[i][n]=sum_val;
}

sum_val=0;
for(i=0;i<n;i++){
    sum_val+=cell_data[n][i];
}
cell_data[n][n]=sum_val;

for(i=0;i<n+1;i++){
    for(j=0;j<n+1;j++){
        printf("%5d", cell_data[i][j]);
    }
    printf("\n");
}
return 0;
}
Copy
Sample Output:

```

Input number of rows/columns:

4

Input the cell value

Row 0 input cell values

25

69

51

26

Row 1 input cell values

68

35

29

54

Row 2 input cell values

54

57

45

63

Row 3 input cell values

61

68

47

59

Result:

```
25 69 51 26 171
68 35 29 54 186
54 57 45 63 219
61 68 47 59 235
208 229 172 202 811
```

3.8

```
#include <stdio.h>
#include <stdlib.h>
char text[128];
int main(void) {
    int i, j, k;
    int result = 0;
    char temp[8];
    printf("Input Sentences with positive integers:\n");
    scanf("%s", text);
    i = 0;
```

```

while (text[i]) {
    for (;  

        (text[i] < '0' || '9' < text[i]) && text[i]; i++);  

    if ('0' <= text[i] && text[i] <= '9') {  

        for (j = 0;  

            '0' <= text[i] && text[i] <= '9'; j++, i++) {  

            temp[j] = text[i];  

        }  

        temp[j] = '\0';  

        result += atoi(temp);  

    }  

}  

printf("\nSum of all numerical values embedded in a sentence:\n");  

printf("%d\n", result);
return 0;
}

```

O/P:

Input Sentences with positive integers:

5littleJackand2mouse.

Sum of all numerical values embedded in a sentence:

7

3.9

```

int findMostFrequentElement(int A[], int n)
{
    sort(A, n)
    int max_freq = 0
    int ans = -1
    int i = 0
    while (i < n)
    {
        int curr_freq = 1
        while (i+1 < n && A[i+1] == A[i])
        {
            curr_freq = curr_freq + 1
            i = i + 1
        }

        if (max_freq < curr_freq)
        {
            max_freq = curr_freq
            ans = A[i]
        }
    }
}

```

```

        else if (max_freq == curr_freq)
            ans = min(ans, A[i])
            i = i + 1
    }
    return ans
}

```

|                   |             |
|-------------------|-------------|
| Assignment        | Unit-4      |
| Assignment Number | 4.1 to 4.10 |
| Date:             |             |

Solution/Answer:

4.1

```

#include <stdio.h>
void main()
{
    int m=300;
    float fx = 300.60;
    char cht = 'z';

    printf("\n\n Pointer : Demonstrate the use of & and * operator :\n");
    printf("-----\n");

```

```

int *pt1;
float *pt2;
char *pt3;
pt1= &m;
pt2=&fx;
pt3=&cht;
printf ( " m = %d\n",m);
printf ( " fx = %f\n",fx);
printf ( " cht = %c\n",cht);
printf("\n Using & operator :\n");
printf("-----\n");
printf ( " address of m = %p\n",&m);
printf ( " address of fx = %p\n",&fx);
printf ( " address of cht = %p\n",&cht);
printf("\n Using & and * operator :\n");
printf("-----\n");
printf ( " value at address of m = %d\n",*(&m));
printf ( " value at address of fx = %f\n",*(&fx));
printf ( " value at address of cht = %c\n",*(&cht));
printf("\n Using only pointer variable :\n");
printf("-----\n");
printf ( " address of m = %p\n",pt1);

```

```
printf ( " address of fx = %p\n",pt2);
printf ( " address of cht = %p\n",pt3);
printf("\n Using only pointer operator :\n");
printf("-----\n");
printf ( " value at address of m = %d\n",*pt1);
printf ( " value at address of fx= %f\n",*pt2);
printf ( " value at address of cht= %c\n\n",*pt3);
}
```

Copy

Sample Output:

Pointer : Demonstrate the use of & and \* operator :

---

```
m = 300
fx = 300.600006
cht = z
```

Using & operator :

---

```
address of m = 0x7fff71cd0b38
address of fx = 0x7fff71cd0b3c
address of cht = 0x7fff71cd0b37
```

Using & and \* operator :

---

```
value at address of m = 300
value at address of fx = 300.600006
value at address of cht = z
```

Using only pointer variable :

---

```
address of m = 0x7fff71cd0b38
address of fx = 0x7fff71cd0b3c
address of cht = 0x7fff71cd0b37
```

Using only pointer operator :

---

```
value at address of m = 300
value at address of fx= 300.600006
value at address of cht= z
```

4.2

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

```

int fno,sno,*ptr1=&fno,*ptr2=&sno;

printf("\n\n Pointer : Find the maximum number between two numbers :\n");
printf("-----\n");

printf(" Input the first number : ");
scanf("%d", ptr1);
printf(" Input the second number : ");
scanf("%d", ptr2);

if(*ptr1>*ptr2)
{
    printf("\n\n %d is the maximum number.\n\n",*ptr1);
}
else
{
    printf("\n\n %d is the maximum number.\n\n",*ptr2);
}

}

```

O/P:

```

Input the first number : 5
Input the second number : 6

```

6 is the maximum number.

4.3

```

#include <stdio.h>
#include <string.h>

void changePosition(char *ch1, char *ch2)
{
    char tmp;
    tmp = *ch1;
    *ch1 = *ch2;
    *ch2 = tmp;
}
void charPermu(char *cht, int stno, int endno)
{
    int i;
    if (stno == endno)
        printf("%s ", cht);
    else
    {

```

```

        for (i = stno; i <= endno; i++)
    {
        changePosition((cht+stno), (cht+i));
        charPermu(cht, stno+1, endno);
        changePosition((cht+stno), (cht+i));
    }
}

int main()
{
    char str[] = "abcd";
    printf("\n\n Pointer : Generate permutations of a given string :\n");
    printf("-----\n");
    int n = strlen(str);
    printf(" The permutations of the string are :\n");
    charPermu(str, 0, n-1);
    printf("\n\n");
    return 0;
}

```

O/P:

The permutations of the string are :

```

abcd abdc acbd acdb adcb adbc bacd badc bcad bcda bdca bdac cbad cbda cabd cadb
cdab cdba dbca dbac dcba dcab dacb dabc

```

#### 4.4

```

#include <stdio.h>
#include <stdlib.h>
int main()
{
    int i,n;
    float *element;
    printf("\n\n Pointer : Find the largest element using Dynamic Memory Allocation :\n");
    printf("-----\n");
    printf(" Input total number of elements(1 to 100): ");
    scanf("%d",&n);
    element=(float*)calloc(n,sizeof(float)); // Memory is allocated for 'n' elements
    if(element==NULL)
    {
        printf(" No memory is allocated.");
        exit(0);
    }
    printf("\n");
    for(i=0;i<n;++i)
    {

```

```

    printf(" Number %d: ",i+1);
    scanf("%f",element+i);
}
for(i=1;i<n;++i)
{
    if(*element<*(element+i))
        *element=*(element+i);
}
printf(" The Largest element is : %.2f \n\n",*element);
return 0;
}

```

Input total number of elements(1 to 100): 5

Number 1: 5  
 Number 2: 7  
 Number 3: 2  
 Number 4: 9  
 Number 5: 8  
 The Largest element is : 9.00

4.5.

```

#include <stdio.h>
int calculateLength(char*);

void main()
{
    char str1[25];
    int l;
    printf("\n\n Pointer : Calculate the length of the string :\n");
    printf("-----\n");

    printf(" Input a string : ");
    fgets(str1, sizeof str1, stdin);

    l = calculateLength(str1);
    printf(" The length of the given string %s is : %d ", str1, l-1);
    printf("\n\n");

}

int calculateLength(char* ch) // ch = base address of array str1 ( &str1[0] )
{
    int ctr = 0;
    while (*ch != '\0')
    {

```

```
    ctr++;
    ch++;
}
return ctr;
}
```

O/P:

```
Input a string : w3resource
The length of the given string w3resource
is : 10
```

4.6

```
#include <stdio.h>
void findFact(int,int*);
int main()
{
    int fact;
    int num1;
    printf("\n\n Pointer :");
    printf("-----");
    printf(" Input a number :");
    scanf("%d",&num1);

    findFact(num1,&fact);
    printf(" The Factorial of %d is : %d \n\n",num1,fact);
    return 0;
}
```

```
void findFact(int n,int *f)
{
    int i;
    *f=1;
    for(i=1;i<=n;i++)
        *f=*f*i;
}
```

O/P:

```
Input a number : 5
The Factorial of 5 is : 120
```

4.7

```
#include <stdio.h>
int main()
```

```

{
    char str1[50];
    char *pt;
    int ctrV,ctrC;
    printf("\n\n Pointer : Count the number of vowels and consonants :\n");
    printf("-----\n");
    printf(" Input a string: ");
    fgets(str1, sizeof str1, stdin);

    //assign address of str1 to pt
    pt=str1;

    ctrV=ctrC=0;
    while(*pt!='\0')
    {
        if(*pt=='A' ||*pt=='E' ||*pt=='I' ||*pt=='O' ||*pt=='U' ||*pt=='a' ||*pt=='e' ||*pt=='i' ||*pt=='o'
        ||*pt=='u')
            ctrV++;
        else
            ctrC++;
        pt++; //pointer is increasing for searching the next character
    }

    printf(" Number of vowels : %d\n Number of consonants : %d\n",ctrV,ctrC-1);
    return 0;
}

```

O/P:

```

Input a string: string
Number of vowels : 1
Number of consonants : 5

```

4.8

```

#include <stdio.h>
void main()
{
    int *a,i,j,tmp,n;
    printf("\n\n Pointer : Sort an array using pointer :\n");
    printf("-----\n");

    printf(" Input the number of elements to store in the array : ");
    scanf("%d",&n);

    printf(" Input %d number of elements in the array : \n",n);
    for(i=0;i<n;i++)
    {

```

```

        printf(" element - %d : ",i+1);
        scanf("%d",a+i);
    }

for(i=0;i<n;i++)
{
    for(j=i+1;j<n;j++)
    {
        if( *(a+i) > *(a+j))
        {
            tmp = *(a+i);
            *(a+i) = *(a+j);
            *(a+j) = tmp;
        }
    }
}

printf("\n The elements in the array after sorting : \n");
for(i=0;i<n;i++)
{
    printf(" element - %d : %d \n",i+1,*(a+i));
}
printf("\n");
}

```

O/P:

Input the number of elements to store in the array : 5

Input 5 number of elements in the array :

element - 1 : 25  
 element - 2 : 45  
 element - 3 : 89  
 element - 4 : 15  
 element - 5 : 82

The elements in the array after sorting :

element - 1 : 15  
 element - 2 : 25  
 element - 3 : 45  
 element - 4 : 82  
 element - 5 : 89

4.9

```
#include <stdio.h>
void main()
{
    int arr1[10];
    int i,n, sum = 0;
    int *pt;
```

```

printf("\n\n Pointer : Sum of all elements in an array :\n");
printf("-----\n");

printf(" Input the number of elements to store in the array (max 10) : ");
scanf("%d",&n);

printf(" Input %d number of elements in the array : \n",n);
for(i=0;i<n;i++)
{
    printf(" element - %d : ",i+1);
    scanf("%d",&arr1[i]);
}

pt = arr1; // pt store the base address of array arr1

for (i = 0; i < n; i++) {
    sum = sum + *pt;
    pt++;
}

printf(" The sum of array is : %d\n\n", sum);
}

```

Copy

Sample Output:

Pointer : Sum of all elements in an array :

---

Input the number of elements to store in the array (max 10) : 5  
Input 5 number of elements in the array :  
element - 1 : 2  
element - 2 : 3  
element - 3 : 4  
element - 4 : 5  
element - 5 : 6  
The sum of array is : 20

4.10

```

#include <stdio.h>
void main()
{
    int n, i, arr1[15];
    int *pt;
    printf("\n\n Pointer : Print the elements of an array in reverse order :\n");
    printf("-----\n");

```

```

printf(" Input the number of elements to store in the array (max 15) : ");
scanf("%d",&n);
pt = &arr1[0]; // pt stores the address of base array arr1
printf(" Input %d number of elements in the array : \n",n);
for(i=0;i<n;i++)
{
    printf(" element - %d : ",i+1);
    scanf("%d",pt);//accept the address of the value);
    pt++;
}
pt = &arr1[n - 1];

printf("\n The elements of array in reverse order are :");

for (i = n; i > 0; i--)
{
    printf("\n element - %d : %d ", i, *pt);
    pt--;
}
printf("\n\n");
}

```

O/P:

Input the number of elements to store in the array (max 15) : 5  
Input 5 number of elements in the array :  
element - 1 : 2  
element - 2 : 3  
element - 3 : 4  
element - 4 : 5  
element - 5 : 6

The elements of array in reverse order are :

element - 5 : 6  
element - 4 : 5  
element - 3 : 4  
element - 2 : 3  
element - 1 : 2

|                           |             |
|---------------------------|-------------|
| Assignment                | Unit-5      |
| Assignment Number         | 5.1 to 5.10 |
| Date                      |             |
| Solution/Answer:          |             |
| 5.1<br>#include <stdio.h> |             |

```

int main ()
{
    FILE * fptr;
    int i,n;
    char str[100];
    char fname[20]="test.txt";
    char str1;

    printf("\n\n Write multiple lines in a text file and read the file :\n");
    printf("-----\n");
    printf(" Input the number of lines to be written : ");
    scanf("%d", &n);
    printf("\n :: The lines are ::\n");
    fptr = fopen (fname,"w");
    for(i = 0; i < n+1;i++)
    {
        fgets(str, sizeof str, stdin);
        fputs(str, fptr);
    }
    fclose (fptr);
/*----- read the file -----*/
    fptr = fopen (fname, "r");
    printf("\n The content of the file %s is :\n",fname);
    str1 = fgetc(fptr);
    while (str1 != EOF)
    {
        printf ("%c", str1);
        str1 = fgetc(fptr);
    }
    printf("\n\n");
    fclose (fptr);
    return 0;
}

```

O/P:

Input the number of lines to be written : 4

:: The lines are ::  
 test line 1  
 test line 2  
 test line 3  
 test line 4

The content of the file test.txt is :

```

test line 1
test line 2
test line 3
test line 4
5.2
#include <stdio.h>

#define FSIZE 100

int main()
{
    FILE *fptr;
    int ctr = 0;
    char fname[FSIZE];
    char c;
    printf("\n\n Read the file and count the number of lines :\n");
    printf("-----\n");
    printf(" Input the file name to be opened : ");
    scanf("%s", fname);

    fptr = fopen(fname, "r");
    if (fptr == NULL)
    {
        printf("Could not open file %s", fname);
        return 0;
    }
    // Extract characters from file and store in character c
    for (c = getc(fptr); c != EOF; c = getc(fptr))
        if (c == '\n') // Increment count if this character is newline
            ctr = ctr + 1;
    fclose(fptr);
    printf(" The lines in the file %s are : %d \n \n", fname, ctr-1);
    return 0;
}

```

O/P:

Input the file name to be opened : test.txt  
The lines in the file test.txt are : 4

5.3

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define LSIZ 128
#define RSIZ 10

```

```

int main(void)
{
    char line[RSIZ][LSIZ];
                                char fname[20];
    FILE *fptr = NULL;
    int i = 0;
    int tot = 0;
    printf("\n\n Find the content of the file and number of lines in a Text File :\n");
    printf("-----\n");
    printf(" Input the file name to be opened : ");
    scanf("%s",fname);

    fptr = fopen(fname, "r");
/*----- store the lines into an array -----*/
    while(fgets(line[i], LSIZ, fptr))
    {
        line[i][strlen(line[i]) - 1] = '\0';
        i++;
    }
    tot = i;
    printf("\n The content of the file %s are : \n",fname);
    for(i = 0; i < tot; ++i)
    {
        printf(" %s\n", line[i]);
    }
/*-----*/
    printf("\n The lines in the file are : %d\n",tot-1);
    printf("\n");

    return 0;
}

```

Copy

Sample Output:

Find the content of the file and number of lines in a Text File :

-----  
Input the file name to be opened : test.txt

The content of the file test.txt are :

test line 1  
 test line 2  
 test line 3  
 test line 4

The lines in the file are : 4

5.4

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

void main()
{
    FILE *fptr;
    char ch;
    int wrd=1,charctr=1;
    char fname[20];
    printf("\n\n Count the number of words and characters in a file :\n");
        printf("-----\n");
        printf(" Input the filename to be opened : ");
        scanf("%s",fname);

    fptr=fopen(fname,"r");
    if(fptr==NULL)
    {
        printf(" File does not exist or can not be opened.");
    }
    else
    {
        ch=fgetc(fptr);
        printf(" The content of the file %s are : ",fname);
        while(ch!=EOF)
        {
            printf("%c",ch);
            if(ch==' '||ch=='\n')
            {
                wrd++;
            }
            else
            {
                charctr++;
            }
            ch=fgetc(fptr);
        }
        printf("\n The number of words in the file %s are : %d\n",fname,wrd-2);
        printf(" The number of characters in the file %s are : %d\n\n",fname,charctr-1);
    }
    fclose(fptr);
}
```

O/P:

Input the filename to be opened : test.txt

The content of the file test.txt are :

test line 1

test line 2

test line 3

test line 4

The number of words in the file test.txt are : 12

The number of characters in the file test.txt are : 36

5.5

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

```
#include <string.h>
```

```
#define MAX 256
```

```
int main()
```

```
{
```

```
    int lno, ctr = 0;
```

```
    char ch;
```

```
    FILE *fptr1, *fptr2;
```

```
    char str[MAX], temp[] = "temp.txt";
```

```
    char fname[MAX];
```

```
specific line from a file :\n");
```

```
printf("\n\n Delete a
```

```
-----\n");
```

```
printf("-----
```

```
name to be opened : ");
```

```
printf(" Input the file
```

```
fptr1 = fopen(fname, "r");
```

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

```
if (!fptr1)
```

```
{
```

```
    printf(" File not found or unable to open the input file!!\n");
```

```
    return 0;
```

```
}
```

```
fptr2 = fopen(temp, "w"); // open the temporary file in write mode
```

```
if (!fptr2)
```

```
{
```

```
    printf("Unable to open a temporary file to write!!\n");
```

```
    fclose(fptr1);
```

```
    return 0;
```

```
}
```

```
printf(" Input the line you want to remove : ");
```

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

```
lno++;
```

```
// copy all contents to the temporary file except the specific line
```

```
while (!feof(fptr1))
```

```

{
    strcpy(str, "\0");
    fgets(str, MAX, fptr1);
    if (!feof(fptr1))
    {
        ctr++;
        /* skip the line at given line number */
        if (ctr != lno)
        {
            fprintf(fptr2, "%s", str);
        }
    }
    fclose(fptr1);
    fclose(fptr2);
    remove(fname); // remove the original file
    rename(temp, fname); // rename the temporary file to original name
/*----- Read the file -----*/
    fptr1=fopen(fname,"r");
    ch=fgetc(fptr1);
    printf(" Now the content of the file %s is : \n",fname);
    while(ch!=EOF)
    {
        printf("%c",ch);
        ch=fgetc(fptr1);
    }
    fclose(fptr1);
/*----- End of reading -----*/
    return 0;
}

```

O/P:

Input the file name to be opened : test.txt

Input the line you want to remove : 2

Now the content of the file test.txt are :

```

test line 1
test line 3
test line 4
5.6
#include <stdio.h>
#include <string.h>

#define MAX 256

```

```

int main()
{
    FILE *fptr1, *fptr2;
    int lno, linectr = 0;
    char str[MAX], fname[MAX];
    char newln[MAX], temp[] = "temp.txt";

    printf("\n\n Replace a
specific line in a text file with a new text :\n");
    printf("-----\n");

    name to be opened : ");
    fgets(fname, MAX, stdin);
    fname[strlen(fname) - 1] = '\0';
    fptr1 = fopen(fname, "r");
    if (!fptr1)
    {
        printf("Unable to open the input file!!\n");
        return 0;
    }
    fptr2 = fopen(temp, "w");
    if (!fptr2)
    {
        printf("Unable to open a temporary file to write!!\n");
        fclose(fptr1);
        return 0;
    }
    /* get the new line from the user */
    printf(" Input the content of the new line : ");
    fgets(newln, MAX, stdin);
    /* get the line number to delete the specific line */
    printf(" Input the line no you want to replace : ");
    scanf("%d", &lno);
    lno++;
    // copy all contents to the temporary file other except specific line
    while (!feof(fptr1))
    {
        strcpy(str, "\0");
        fgets(str, MAX, fptr1);
        if (!feof(fptr1))
        {
            linectr++;
            if (linectr != lno)
            {
                fprintf(fptr2, "%s", str);
            }
        }
    }
}

```

```

        }
    else
    {
        fprintf(fptr2, "%s", newln);
    }
}
fclose(fptr1);
fclose(fptr2);
remove(fname);
rename(temp, fname);
printf(" Replacement did successfully..!! \n");
return 0;
}

```

O/P:

```

Input the file name to be opened : test.txt
Input the content of the new line : 2
Input the line no you want to replace : 2
Replacement did successfully..!!

```

5.7

#include <stdio.h>

```

int main ()
{
    FILE * fptr;
    int i,n;
    char str[100];
    char fname[20];
    char str1;

    printf("\n\n Append multiple lines at the end of a text file :\n");
    printf("-----\n");
    printf(" Input the file name to be opened : ");
    scanf("%s",fname);

    fptr = fopen(fname, "a");
    printf(" Input the number of lines to be written : ");
    scanf("%d", &n);
    printf(" The lines are : \n");
    for(i = 0; i < n+1;i++)
    {
        fgets(str, sizeof str, stdin);
        fputs(str, fptr);
    }
    fclose (fptr);
//---- Read the file after appended -----

```

```

        fptr = fopen (fname, "r");
        printf("\n The content of the file %s is :\n", fname);
        str1 = fgetc(fptr);
        while (str1 != EOF)
        {
            printf ("%c", str1);
            str1 = fgetc(fptr);
        }
        printf("\n\n");
        fclose (fptr);
//----- End of reading -----
        return 0;
    }
}

```

O/P:

```

Input the file name to be opened : test.txt
Input the number of lines to be written : 3
The lines are :
test line 5
test line 6
test line 7

```

The content of the file test.txt is :

```

test line 1
test line 2
test line 3
test line 4

test line 5
test line 6
test line 7
5.8
#include <stdio.h>
#include <stdlib.h>

void main()
{
FILE *fptr1, *fptr2;
char ch, fname1[20], fname2[20];
printf("\n\n Copy a file in another name :\n");
printf("-----\n");

printf(" Input the source file name : ");
scanf("%s", fname1);
fptr1=fopen(fname1, "r");

```

```

if(fptr1==NULL)
{
printf(" File does not found or error in opening.!!");
exit(1);
}
printf(" Input the new file name : ");
scanf("%s",fname2);
fptr2=fopen(fname2, "w");
if(fptr2==NULL)
{
printf(" File does not found or error in opening.!!");
fclose(fptr1);
exit(2);
}
while(1)
{
ch=fgetc(fptr1);
if(ch==EOF)
{
break;
}
else
{
fputc(ch, fptr2);
}
}
printf(" The file %s copied successfully in the file %s. \n\n",fname1,fname2);
fclose(fptr1);
fclose(fptr2);
getchar();
}

```

O/P:

```

Input the source file name : test.txt
Input the new file name : test1.txt
The file test.txt copied successfully in the file test1.txt.

```

5.9

```

#include <stdio.h>
#include <stdlib.h>

void main()
{
FILE *fold1, *fold2, *fnew;
char ch, fname1[20], fname2[20], fname3[30];
printf("\n\n Merge two files and write it in a new file :\n");

```

```

printf("-----\n");

printf(" Input the 1st file name : ");
scanf("%s",fname1);
printf(" Input the 2nd file name : ");
scanf("%s",fname2);
printf(" Input the new file name where to merge the above two files : ");
scanf("%s",fname3);
fold1=fopen(fname1, "r");
fold2=fopen(fname2, "r");
if(fold1==NULL || fold2==NULL)
{
// perror("Error Message ");
printf(" File does not exist or error in opening...!!\n");
exit(EXIT_FAILURE);
}
fnew=fopen(fname3, "w");
if(fnew==NULL)
{
// perror("Error Message ");
printf(" File does not exist or error in opening...!!\n");
exit(EXIT_FAILURE);
}
while((ch=fgetc(fold1))!=EOF)
{
fputc(ch, fnew);
}
while((ch=fgetc(fold2))!=EOF)
{
fputc(ch, fnew);
}
printf(" The two files merged into %s file successfully..!!\n\n", fname3);
fclose(fold1);
fclose(fold2);
fclose(fnew);
}

```

O/P:

```

Input the 1st file name : test.txt
Input the 2nd file name : test1.txt
Input the new file name where to merge the above two files : mergefiles.txt
The two files merged into mergefiles.txt file successfully..!!

```

5.10

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

```
void main()
{
int status;
char fname[20];
printf("\n\n Remove a file from the disk :\n");
printf("-----\n");
printf(" Input the name of file to delete : ");
scanf("%s",fname);
status=remove(fname);
if(status==0)
{
    printf(" The file %s is deleted successfully..!!\n\n",fname);
}
else
{
    printf(" Unable to delete file %s\n\n",fname);
}
}
```

O/P:

Input the name of file to delete : test.txt  
The file test.txt is deleted successfully..!!

**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)**  
 Gandipet, Hyderabad -75

**B.E. (CSE), I Semester Class Test-I, January 2021**

**SUBJECT: Programming for Problem Solving**

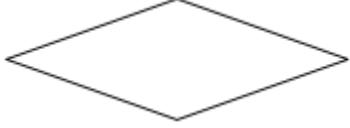
**CODE:20CS C01**

**Time: 30minutes**

**Max. Marks – 20**

| Unit – I |                                                                                                                                                                                                                                                                                                                                                          |       |         |    |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|----|
| S.No.    | Question                                                                                                                                                                                                                                                                                                                                                 | Marks | CO      | BT |
| 1        | <p>Recall high level language source program is translated to object code file by----- and object code file is translated into machine target program by ----- translators.</p> <p>A. Interpreter, compiler<br/>         B. Compiler, Interpreter<br/>         C. Compiler, Assembler<br/>         D. Assembler, Interpreter</p> <p><b>Answer: C</b></p> | 1     | CO<br>1 | L1 |
| 2        | <p>Identify, CPU components-----</p> <p>A. control unit , ALU and register array<br/>         B. control unit , ALU and register array, input unit<br/>         C. control unit , ALU and register array, input and output unit<br/>         D. control unit , ALU and register array, secondary memory</p> <p><b>Answer: A</b></p>                      | 1     | CO<br>1 | L3 |
| 3        | <p>Which software, is the layer between application software and the machine:</p> <p>A. System software<br/>         B. Device drivers<br/>         C. Compiler<br/>         D. Integrated Development Environment</p> <p><b>Answer: A</b></p>                                                                                                           | 1     | CO<br>1 | L1 |
| 4        | <p>For -----generation programming language, programmer need not provide programming logic, because language has built-in logic for the constructs</p> <p>A. 4GL<br/>         B. 3GL<br/>         C. 5GL<br/>         D. 1GL</p> <p><b>Answer: A</b></p>                                                                                                 | 1     | CO<br>1 | L2 |
| 5        | <p>Choose, -----is an implementation of an algorithm, to be run on a specific computer and operating system</p> <p>A. Pseudocode<br/>         B. Flowchart<br/>         C. Program<br/>         D. Instruction</p> <p><b>Answer: C</b></p>                                                                                                               | 1     | CO<br>2 | L1 |
| 6        | <p>Which of the following is not a valid C variable name?</p> <p>A. int number;</p>                                                                                                                                                                                                                                                                      | 1     | CO<br>2 | L2 |

|    |                                                                                                                                                                                                                                                              |   |         |    |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---------|----|
|    | B. float rate;<br>C. int variable_count;<br>D. int \$main;<br><b>Answer: D</b>                                                                                                                                                                               |   |         |    |
| 7  | what is the output of the following program?<br><pre>#include&lt;stdio.h&gt; int main() {     int a = 5, b = 10;     int c;     c = a * 2 + b/2;     printf("\n output = %d", c);     return 0; }</pre> a. 10<br>b. 15<br>c. 20<br>d. 35<br><b>Answer: b</b> | 1 | CO<br>3 | L3 |
| 8  | what will be the output of following code?<br><pre>#include &lt;stdio.h&gt; int main() {     int a = 2;     int b = a++;     printf("%d %d", b, a); }</pre> A. 3 4<br>B. 4 4<br>C. 3 5<br>D. 2 3<br><b>Answer: D</b>                                         | 1 | CO<br>3 | L3 |
| 9  | Which operators are used to compare the values of operands to produce logical value in C language?<br>A. Relational operator<br>B. Logical operator<br>C. Assignment operator<br>D. Arithmetic operator<br><br><b>Answer: A</b>                              | 1 | CO<br>3 | L2 |
| 10 | The following box denotes?                                                                                                                                                                                                                                   | 1 | CO<br>2 | L2 |

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |      |     |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|------|-----|
|    |  <p>A. Decision<br/>B. Initiation<br/>C. Initialization<br/>D. I/O</p> <p><b>Answer: A</b></p>                                                                                                                                                                                                                                                                  |   |      |     |
| 11 | <p>Select the scenario in which if-else statement types are better than switch statement.</p> <p>A. Need to take decision by testing multiple conditions<br/>B. Decision depends on the comparison of 2 different variable data.<br/>C. test in a range of values<br/>D. All of these</p> <p><b>Answer: D. All of these</b></p>                                                                                                                  | 1 | CO 3 | BT1 |
| 12 | <p>What is the output of the following code?</p> <p><b>Line No.</b></p> <pre> 1 #include &lt;stdio.h&gt; 2 void main() 3 { 4     int option = 0; 5     switch (option) 6     { 7         case '0': printf("CSE"); 8             break; 9         case '1': printf("ECE"); 10            break; 11     default: printf("EEE"); 12 } 13 } 14 }</pre> <p>A. Error in the code<br/>B. CSE<br/>C. ECE<br/>D. EEE</p> <p><b>Answer: D. EEE</b></p>     | 1 | CO 3 | BT1 |
| 13 | <p>Recall the control statements concept and tell the following statements are TRUE or FALSE.</p> <p>Statement-X: “else block is optional in if-else if ladder control statement”<br/>Statement-Y: “else block is optional in if-else control statement”</p> <p>A. Statement-X is TRUE and Statement-Y is FALSE<br/>B. Statement-X is FALSE and Statement-Y is TRUE<br/>C. Both the Statements are TRUE<br/>D. Both the Statements are FALSE</p> | 1 | CO 3 | BT1 |

|    |                                                                                                                                                                                                                                                                                                                   |   |      |     |
|----|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|------|-----|
|    | Answer: C. Both the Statements are TRUE                                                                                                                                                                                                                                                                           |   |      |     |
| 14 | <p>What is the output of C program with functions?</p> <pre>void show();</pre> <pre>int main() {     show();     printf("COLLEGE ");     return 0; }</pre> <pre>void show() {     printf("CBIT "); }</pre> <p>A.COLLEGE CBIT<br/> B.CBIT COLLEGE<br/> C.COLLEGE<br/> D.Compiler error</p> <p><b>ANSWER: B</b></p> | 1 | CO 4 | BT3 |
| 15 | <p>How many values can a C Function return at a time.?</p> <p>A. Only One Value<br/> B. Maximum of two values<br/> C. Maximum of three values<br/> D. Maximum of 8 values</p> <p><b>ANSWER:A</b></p>                                                                                                              | 1 | CO 1 | BL2 |
| 16 | <p>How many times Hello is printed</p> <pre>#include&lt;stdio.h&gt; int main() {     int i = -5;     while (i &lt;= 5)     {         if (i &gt;= 0)             break;         else         {             i++;             continue;         }         printf("Hello");     } }</pre>                             | 1 | CO 3 | BL3 |

|    |                                                                                                                                                                                                                            |   |      |     |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|------|-----|
|    | <pre>        return 0;     } A. 10 times B. 5 times C. Infinite times D. 0 times</pre> <p><b>ANSWER:D</b></p>                                                                                                              |   |      |     |
| 17 | <p>What is the output of the program?</p> <pre>#include&lt;stdio.h&gt; static int k;  int main() {     printf("%d", k);      return 0; }</pre> <p>A) -1<br/>B) 0<br/>C) 90<br/>D) Compiler error</p> <p><b>Ans : B</b></p> | 1 | CO 2 | BT1 |
| 18 | <p>A recursive function without If and Else conditions will always lead to?</p> <p>A) Finite loop<br/>B) Infinite loop<br/>C) Incorrect result<br/>D) Correct result</p> <p><b>Ans : B</b></p>                             | 1 | CO 4 | BT2 |
| 19 | <p>Which variable has the longest scope?</p> <pre>#include &lt;stdio.h&gt; float a; int b; int main() {     int c;     return 0; }</pre> <p>A. a<br/>B. b<br/>C. c<br/>D. Both a and b</p> <p><b>Answer: d</b></p>         | 1 | CO 2 | BT1 |
| 20 | <p>Actual and formal parameters must agree in</p> <p>A. Data types<br/>B. Number of arguments and Data types<br/>C. Names and Data type<br/>D. None</p>                                                                    | 1 | CO 4 | BT2 |

|  |                 |  |  |  |  |
|--|-----------------|--|--|--|--|
|  | <b>ANSWER:B</b> |  |  |  |  |
|--|-----------------|--|--|--|--|

**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)**  
 Gandipet, Hyderabad -75

**B.E. (CSE), I Semester Class Test-I, January 2021**

**SUBJECT: Programming for Problem Solving**

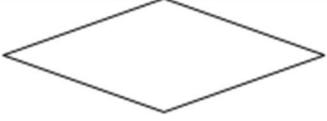
**CODE:20CS C01**

**Time: 30minutes**

**Max. Marks – 20**

| Unit – I |                                                                                                                                                                                                                                                                                                                                                          |       |      |    |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|------|----|
| S.No.    | Question                                                                                                                                                                                                                                                                                                                                                 | Marks | CO   | BT |
| 1        | <p>Recall high level language source program is translated to object code file by----- and object code file is translated into machine target program by ----- translators.</p> <p>A. Interpreter, compiler<br/>         B. Compiler, Interpreter<br/>         C. Compiler, Assembler<br/>         D. Assembler, Interpreter</p> <p><b>Answer: C</b></p> | 1     | CO 1 | L1 |
| 2        | <p>Identify, CPU components-----</p> <p>A. control unit , ALU and register array<br/>         B. control unit , ALU and register array, input unit<br/>         C. control unit , ALU and register array, input and output unit<br/>         D. control unit , ALU and register array, secondary memory</p> <p><b>Answer: A</b></p>                      | 1     | CO 1 | L3 |
| 3        | <p>Which software, is the layer between application software and the machine:</p> <p>A. System software<br/>         B. Device drivers<br/>         C. Compiler<br/>         D. Integrated Development Environment</p> <p><b>Answer: A</b></p>                                                                                                           | 1     | CO 1 | L1 |
| 4        | <p>For -----generation programming language, programmer need not provide programming logic, because language has built-in logic for the constructs</p> <p>A. 4GL<br/>         B. 3GL<br/>         C. 5GL<br/>         D. 1GL</p> <p><b>Answer: A</b></p>                                                                                                 | 1     | CO 1 | L2 |
| 5        | <p>Choose, -----is an implementation of an algorithm, to be run on a specific computer and operating system</p> <p>A. Pseudocode<br/>         B. Flowchart<br/>         C. Program<br/>         D. Instruction</p> <p><b>Answer: C</b></p>                                                                                                               | 1     | CO 2 | L1 |
| 6        | <p>Which of the following is not a valid C variable name?</p> <p>A. int number;</p>                                                                                                                                                                                                                                                                      | 1     | CO 2 | L2 |

|    |                                                                                                                                                                                                                                                              |   |         |    |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---------|----|
|    | B. float rate;<br>C. int variable_count;<br>D. int \$main;<br><b>Answer: D</b>                                                                                                                                                                               |   |         |    |
| 7  | what is the output of the following program?<br><pre>#include&lt;stdio.h&gt; int main() {     int a = 5, b = 10;     int c;     c = a * 2 + b/2;     printf("\n output = %d", c);     return 0; }</pre> a. 10<br>b. 15<br>c. 20<br>d. 35<br><b>Answer: b</b> | 1 | CO<br>3 | L3 |
| 8  | what will be the output of following code?<br><pre>#include &lt;stdio.h&gt; int main() {     int a = 2;     int b = a++;     printf("%d %d", b, a); }</pre> A. 3 4<br>B. 4 4<br>C. 3 5<br>D. 2 3<br><b>Answer: D</b>                                         | 1 | CO<br>3 | L3 |
| 9  | Which operators are used to compare the values of operands to produce logical value in C language?<br>A. Relational operator<br>B. Logical operator<br>C. Assignment operator<br>D. Arithmetic operator<br><br><b>Answer: A</b>                              | 1 | CO<br>3 | L2 |
| 10 | The following box denotes?                                                                                                                                                                                                                                   | 1 | CO<br>2 | L2 |

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |      |     |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|------|-----|
|    |  <p>A. Decision<br/>B. Initiation<br/>C. Initialization<br/>D. I/O</p> <p><b>Answer: A</b></p>                                                                                                                                                                                                                                                                  |   |      |     |
| 11 | <p>Select the scenario in which if-else statement types are better than switch statement.</p> <p>A. Need to take decision by testing multiple conditions<br/>B. Decision depends on the comparison of 2 different variable data.<br/>C. test in a range of values<br/>D. All of these</p> <p><b>Answer: D. All of these</b></p>                                                                                                                  | 1 | CO 3 | BT1 |
| 12 | <p>What is the output of the following code?</p> <p>Line No.</p> <pre> 1 #include &lt;stdio.h&gt; 2 void main() 3 { 4     int option = 0; 5     switch (option) 6 { 7         case '0': printf("CSE"); 8             break; 9         case '1': printf("ECE"); 10            break; 11     default: printf("EEE"); 12 } 13 14 }</pre> <p>A. Error in the code<br/>B. CSE<br/>C. ECE<br/>D. EEE</p> <p><b>Answer: D. EEE</b></p>                  | 1 | CO 3 | BT1 |
| 13 | <p>Recall the control statements concept and tell the following statements are TRUE or FALSE.</p> <p>Statement-X: “else block is optional in if-else if ladder control statement”<br/>Statement-Y: “else block is optional in if-else control statement”</p> <p>A. Statement-X is TRUE and Statement-Y is FALSE<br/>B. Statement-X is FALSE and Statement-Y is TRUE<br/>C. Both the Statements are TRUE<br/>D. Both the Statements are FALSE</p> | 1 | CO 3 | BT1 |

|    |                                                                                                                                                                                                                 |   |         |     |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---------|-----|
|    | Answer: C. Both the Statements are TRUE                                                                                                                                                                         |   |         |     |
|    | What is the output of C program with functions?<br>void show();<br><br>int main()<br>{<br>show();<br>printf("COLLEGE ");<br>return 0;<br>}<br><br>14 void show()<br>{<br>printf("CBIT ");<br>}                  | 1 | CO<br>4 | BT3 |
| 14 | A.COLLEGE CBIT<br>B.CBIT COLLEGE<br>C.COLLEGE<br>D.Compiler error                                                                                                                                               |   |         |     |
|    | <b>ANSWER: B</b>                                                                                                                                                                                                |   |         |     |
| 15 | How many values can a C Function return at a time.?<br><br>A. Only One Value<br>B. Maximum of two values<br>C. Maximum of three values<br>D. Maximum of 8 values                                                | 1 | CO<br>1 | BL2 |
|    | <b>ANSWER:A</b>                                                                                                                                                                                                 |   |         |     |
| 16 | How many times Hello is printed<br><br>#include<stdio.h><br>int main()<br>{<br>int i = -5;<br>while (i <= 5)<br>{<br>if (i >= 0)<br>break;<br>else<br>{<br>i++;<br>continue;<br>}<br>printf("Hello");<br>}<br>} | 1 | CO<br>3 | BL3 |

|    |                                                                                                                                                                                                                            |   |      |     |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|------|-----|
|    | <pre>         return 0;     } A. 10 times B. 5 times C. Infinite times D. 0 times <b>ANSWER:D</b> </pre>                                                                                                                   |   |      |     |
| 17 | <p>What is the output of the program?</p> <pre>#include&lt;stdio.h&gt; static int k;  int main() {     printf("%d", k);      return 0; }</pre> <p>A) -1<br/>B) 0<br/>C) 90<br/>D) Compiler error</p> <p><b>Ans : B</b></p> | 1 | CO 2 | BT1 |
| 18 | <p>A recursive function without If and Else conditions will always lead to?</p> <p>A) Finite loop<br/>B) Infinite loop<br/>C) Incorrect result<br/>D) Correct result</p> <p><b>Ans : B</b></p>                             | 1 | CO 4 | BT2 |
| 19 | <p>Which variable has the longest scope?</p> <pre>#include &lt;stdio.h&gt; float a; int b; int main() {     int c;     return 0; }</pre> <p>A. a<br/>B. b<br/>C. c<br/>D. Both a and b</p> <p><b>Answer: d</b></p>         | 1 | CO 2 | BT1 |
| 20 | <p>Actual and formal parameters must agree in</p> <p>A. Data types<br/>B. Number of arguments and Data types<br/>C. Names and Data type<br/>D. None</p>                                                                    | 1 | CO 4 | BT2 |

|  |                 |  |  |  |
|--|-----------------|--|--|--|
|  | <b>ANSWER:B</b> |  |  |  |
|--|-----------------|--|--|--|

**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)**  
 Gandipet, Hyderabad -75

**B.E. (CSE), I Semester Class Test-I, January 2021**

**SUBJECT: Programming for Problem Solving**

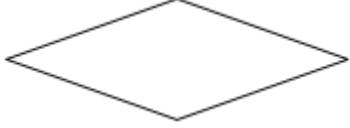
**CODE:20CS C01**

**Time: 30minutes**

**Max. Marks – 20**

| Unit – I |                                                                                                                                                                                                                                                                                                                                                          |       |      |    |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|------|----|
| S.No.    | Question                                                                                                                                                                                                                                                                                                                                                 | Marks | CO   | BT |
| 1        | <p>Recall high level language source program is translated to object code file by----- and object code file is translated into machine target program by ----- translators.</p> <p>A. Interpreter, compiler<br/>         B. Compiler, Interpreter<br/>         C. Compiler, Assembler<br/>         D. Assembler, Interpreter</p> <p><b>Answer: C</b></p> | 1     | CO 1 | L1 |
| 2        | <p>Identify, CPU components-----</p> <p>A. control unit , ALU and register array<br/>         B. control unit , ALU and register array, input unit<br/>         C. control unit , ALU and register array, input and output unit<br/>         D. control unit , ALU and register array, secondary memory</p> <p><b>Answer: A</b></p>                      | 1     | CO 1 | L3 |
| 3        | <p>Which software, is the layer between application software and the machine:</p> <p>A. System software<br/>         B. Device drivers<br/>         C. Compiler<br/>         D. Integrated Development Environment</p> <p><b>Answer: A</b></p>                                                                                                           | 1     | CO 1 | L1 |
| 4        | <p>For -----generation programming language, programmer need not provide programming logic, because language has built-in logic for the constructs</p> <p>A. 4GL<br/>         B. 3GL<br/>         C. 5GL<br/>         D. 1GL</p> <p><b>Answer: A</b></p>                                                                                                 | 1     | CO 1 | L2 |
| 5        | <p>Choose, -----is an implementation of an algorithm, to be run on a specific computer and operating system</p> <p>A. Pseudocode<br/>         B. Flowchart<br/>         C. Program<br/>         D. Instruction</p> <p><b>Answer: C</b></p>                                                                                                               | 1     | CO 2 | L1 |
| 6        | <p>Which of the following is not a valid C variable name?</p> <p>A. int number;</p>                                                                                                                                                                                                                                                                      | 1     | CO 2 | L2 |

|    |                                                                                                                                                                                                                                                              |   |         |    |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---------|----|
|    | B. float rate;<br>C. int variable_count;<br>D. int \$main;<br><b>Answer: D</b>                                                                                                                                                                               |   |         |    |
| 7  | what is the output of the following program?<br><pre>#include&lt;stdio.h&gt; int main() {     int a = 5, b = 10;     int c;     c = a * 2 + b/2;     printf("\n output = %d", c);     return 0; }</pre> a. 10<br>b. 15<br>c. 20<br>d. 35<br><b>Answer: b</b> | 1 | CO<br>3 | L3 |
| 8  | what will be the output of following code?<br><pre>#include &lt;stdio.h&gt; int main() {     int a = 2;     int b = a++;     printf("%d %d", b, a); }</pre> A. 3 4<br>B. 4 4<br>C. 3 5<br>D. 2 3<br><b>Answer: D</b>                                         | 1 | CO<br>3 | L3 |
| 9  | Which operators are used to compare the values of operands to produce logical value in C language?<br>A. Relational operator<br>B. Logical operator<br>C. Assignment operator<br>D. Arithmetic operator<br><br><b>Answer: A</b>                              | 1 | CO<br>3 | L2 |
| 10 | The following box denotes?                                                                                                                                                                                                                                   | 1 | CO<br>2 | L2 |

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |      |     |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|------|-----|
|    |  <p>A. Decision<br/>B. Initiation<br/>C. Initialization<br/>D. I/O</p> <p><b>Answer: A</b></p>                                                                                                                                                                                                                                                                  |   |      |     |
| 11 | <p>Select the scenario in which if-else statement types are better than switch statement.</p> <p>A. Need to take decision by testing multiple conditions<br/>B. Decision depends on the comparison of 2 different variable data.<br/>C. test in a range of values<br/>D. All of these</p> <p><b>Answer: D. All of these</b></p>                                                                                                                  | 1 | CO 3 | BT1 |
| 12 | <p>What is the output of the following code?</p> <p><b>Line No.</b></p> <pre> 1 #include &lt;stdio.h&gt; 2 void main() 3 { 4     int option = 0; 5     switch (option) 6     { 7         case '0': printf("CSE"); 8             break; 9         case '1': printf("ECE"); 10            break; 11     default: printf("EEE"); 12 } 13 } 14 }</pre> <p>A. Error in the code<br/>B. CSE<br/>C. ECE<br/>D. EEE</p> <p><b>Answer: D. EEE</b></p>     | 1 | CO 3 | BT1 |
| 13 | <p>Recall the control statements concept and tell the following statements are TRUE or FALSE.</p> <p>Statement-X: “else block is optional in if-else if ladder control statement”<br/>Statement-Y: “else block is optional in if-else control statement”</p> <p>A. Statement-X is TRUE and Statement-Y is FALSE<br/>B. Statement-X is FALSE and Statement-Y is TRUE<br/>C. Both the Statements are TRUE<br/>D. Both the Statements are FALSE</p> | 1 | CO 3 | BT1 |

|    |                                                                                                                                                                                                                                                                                                                |   |      |     |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|------|-----|
|    | Answer: C. Both the Statements are TRUE                                                                                                                                                                                                                                                                        |   |      |     |
| 14 | <p>What is the output of C program with functions?</p> <pre>void show();</pre> <pre>int main() {     show();     printf("COLLEGE ");     return 0; }</pre> <pre>void show() {     printf("CBIT "); }</pre> <p>A.COLLEGE CBIT<br/>B.CBIT COLLEGE<br/>C.COLLEGE<br/>D.Compiler error</p> <p><b>ANSWER: B</b></p> | 1 | CO 4 | BT3 |
| 15 | <p>How many values can a C Function return at a time.?</p> <p>A. Only One Value<br/>B. Maximum of two values<br/>C. Maximum of three values<br/>D. Maximum of 8 values</p> <p><b>ANSWER:A</b></p>                                                                                                              | 1 | CO 1 | BL2 |
| 16 | <p>How many times Hello is printed</p> <pre>#include&lt;stdio.h&gt; int main() {     int i = -5;     while (i &lt;= 5)     {         if (i &gt;= 0)             break;         else         {             i++;             continue;         }         printf("Hello");     } }</pre>                          | 1 | CO 3 | BL3 |

|    |                                                                                                                                                                                                                            |   |      |     |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|------|-----|
|    | <pre>        return 0;     } A. 10 times B. 5 times C. Infinite times D. 0 times</pre> <p><b>ANSWER:D</b></p>                                                                                                              |   |      |     |
| 17 | <p>What is the output of the program?</p> <pre>#include&lt;stdio.h&gt; static int k;  int main() {     printf("%d", k);      return 0; }</pre> <p>A) -1<br/>B) 0<br/>C) 90<br/>D) Compiler error</p> <p><b>Ans : B</b></p> | 1 | CO 2 | BT1 |
| 18 | <p>A recursive function without If and Else conditions will always lead to?</p> <p>A) Finite loop<br/>B) Infinite loop<br/>C) Incorrect result<br/>D) Correct result</p> <p><b>Ans : B</b></p>                             | 1 | CO 4 | BT2 |
| 19 | <p>Which variable has the longest scope?</p> <pre>#include &lt;stdio.h&gt; float a; int b; int main() {     int c;     return 0; }</pre> <p>A. a<br/>B. b<br/>C. c<br/>D. Both a and b</p> <p><b>Answer: d</b></p>         | 1 | CO 2 | BT1 |
| 20 | <p>Actual and formal parameters must agree in</p> <p>A. Data types<br/>B. Number of arguments and Data types<br/>C. Names and Data type<br/>D. None</p>                                                                    | 1 | CO 4 | BT2 |

|  |                 |  |  |  |  |
|--|-----------------|--|--|--|--|
|  | <b>ANSWER:B</b> |  |  |  |  |
|--|-----------------|--|--|--|--|

**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)**  
 Gandipet, Hyderabad -75

**B.E. (CSE), I Semester Class Test-I, January 2021**

**SUBJECT: Programming for Problem Solving**

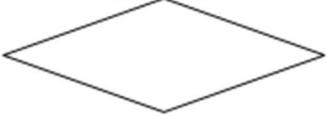
**CODE:20CS C01**

**Time: 30minutes**

**Max. Marks – 20**

| Unit – I |                                                                                                                                                                                                                                                                                                                                                          |       |         |    |
|----------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|---------|----|
| S.No.    | Question                                                                                                                                                                                                                                                                                                                                                 | Marks | CO      | BT |
| 1        | <p>Recall high level language source program is translated to object code file by----- and object code file is translated into machine target program by ----- translators.</p> <p>A. Interpreter, compiler<br/>         B. Compiler, Interpreter<br/>         C. Compiler, Assembler<br/>         D. Assembler, Interpreter</p> <p><b>Answer: C</b></p> | 1     | CO<br>1 | L1 |
| 2        | <p>Identify, CPU components-----</p> <p>A. control unit , ALU and register array<br/>         B. control unit , ALU and register array, input unit<br/>         C. control unit , ALU and register array, input and output unit<br/>         D. control unit , ALU and register array, secondary memory</p> <p><b>Answer: A</b></p>                      | 1     | CO<br>1 | L3 |
| 3        | <p>Which software, is the layer between application software and the machine:</p> <p>A. System software<br/>         B. Device drivers<br/>         C. Compiler<br/>         D. Integrated Development Environment</p> <p><b>Answer: A</b></p>                                                                                                           | 1     | CO<br>1 | L1 |
| 4        | <p>For -----generation programming language, programmer need not provide programming logic, because language has built-in logic for the constructs</p> <p>A. 4GL<br/>         B. 3GL<br/>         C. 5GL<br/>         D. 1GL</p> <p><b>Answer: A</b></p>                                                                                                 | 1     | CO<br>1 | L2 |
| 5        | <p>Choose, -----is an implementation of an algorithm, to be run on a specific computer and operating system</p> <p>A. Pseudocode<br/>         B. Flowchart<br/>         C. Program<br/>         D. Instruction</p> <p><b>Answer: C</b></p>                                                                                                               | 1     | CO<br>2 | L1 |
| 6        | <p>Which of the following is not a valid C variable name?</p> <p>A. int number;</p>                                                                                                                                                                                                                                                                      | 1     | CO<br>2 | L2 |

|    |                                                                                                                                                                                                                                                              |   |         |    |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---------|----|
|    | B. float rate;<br>C. int variable_count;<br>D. int \$main;<br><b>Answer: D</b>                                                                                                                                                                               |   |         |    |
| 7  | what is the output of the following program?<br><pre>#include&lt;stdio.h&gt; int main() {     int a = 5, b = 10;     int c;     c = a * 2 + b/2;     printf("\n output = %d", c);     return 0; }</pre> a. 10<br>b. 15<br>c. 20<br>d. 35<br><b>Answer: b</b> | 1 | CO<br>3 | L3 |
| 8  | what will be the output of following code?<br><pre>#include &lt;stdio.h&gt; int main() {     int a = 2;     int b = a++;     printf("%d %d", b, a); }</pre> A. 3 4<br>B. 4 4<br>C. 3 5<br>D. 2 3<br><b>Answer: D</b>                                         | 1 | CO<br>3 | L3 |
| 9  | Which operators are used to compare the values of operands to produce logical value in C language?<br>A. Relational operator<br>B. Logical operator<br>C. Assignment operator<br>D. Arithmetic operator<br><br><b>Answer: A</b>                              | 1 | CO<br>3 | L2 |
| 10 | The following box denotes?                                                                                                                                                                                                                                   | 1 | CO<br>2 | L2 |

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                  |   |      |     |
|----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|------|-----|
|    |  <p>A. Decision<br/>B. Initiation<br/>C. Initialization<br/>D. I/O</p> <p><b>Answer: A</b></p>                                                                                                                                                                                                                                                                  |   |      |     |
| 11 | <p>Select the scenario in which if-else statement types are better than switch statement.</p> <p>A. Need to take decision by testing multiple conditions<br/>B. Decision depends on the comparison of 2 different variable data.<br/>C. test in a range of values<br/>D. All of these</p> <p><b>Answer: D. All of these</b></p>                                                                                                                  | 1 | CO 3 | BT1 |
| 12 | <p>What is the output of the following code?</p> <p>Line No.</p> <pre> 1 #include &lt;stdio.h&gt; 2 void main() 3 { 4     int option = 0; 5     switch (option) 6 { 7         case '0': printf("CSE"); 8             break; 9         case '1': printf("ECE"); 10            break; 11     default: printf("EEE"); 12 } 13 14 }</pre> <p>A. Error in the code<br/>B. CSE<br/>C. ECE<br/>D. EEE</p> <p><b>Answer: D. EEE</b></p>                  | 1 | CO 3 | BT1 |
| 13 | <p>Recall the control statements concept and tell the following statements are TRUE or FALSE.</p> <p>Statement-X: “else block is optional in if-else if ladder control statement”<br/>Statement-Y: “else block is optional in if-else control statement”</p> <p>A. Statement-X is TRUE and Statement-Y is FALSE<br/>B. Statement-X is FALSE and Statement-Y is TRUE<br/>C. Both the Statements are TRUE<br/>D. Both the Statements are FALSE</p> | 1 | CO 3 | BT1 |

|    |                                                                                                                                                                                                                 |   |         |     |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|---------|-----|
|    | Answer: C. Both the Statements are TRUE                                                                                                                                                                         |   |         |     |
|    | What is the output of C program with functions?<br>void show();<br><br>int main()<br>{<br>show();<br>printf("COLLEGE ");<br>return 0;<br>}<br><br>14 void show()<br>{<br>printf("CBIT ");<br>}                  | 1 | CO<br>4 | BT3 |
| 14 | A.COLLEGE CBIT<br>B.CBIT COLLEGE<br>C.COLLEGE<br>D.Compiler error                                                                                                                                               |   |         |     |
|    | <b>ANSWER: B</b>                                                                                                                                                                                                |   |         |     |
| 15 | How many values can a C Function return at a time.?<br><br>A. Only One Value<br>B. Maximum of two values<br>C. Maximum of three values<br>D. Maximum of 8 values                                                | 1 | CO<br>1 | BL2 |
|    | <b>ANSWER:A</b>                                                                                                                                                                                                 |   |         |     |
| 16 | How many times Hello is printed<br><br>#include<stdio.h><br>int main()<br>{<br>int i = -5;<br>while (i <= 5)<br>{<br>if (i >= 0)<br>break;<br>else<br>{<br>i++;<br>continue;<br>}<br>printf("Hello");<br>}<br>} | 1 | CO<br>3 | BL3 |

|    |                                                                                                                                                                                                                            |   |      |     |
|----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---|------|-----|
|    | <pre>         return 0;     } A. 10 times B. 5 times C. Infinite times D. 0 times <b>ANSWER:D</b> </pre>                                                                                                                   |   |      |     |
| 17 | <p>What is the output of the program?</p> <pre>#include&lt;stdio.h&gt; static int k;  int main() {     printf("%d", k);      return 0; }</pre> <p>A) -1<br/>B) 0<br/>C) 90<br/>D) Compiler error</p> <p><b>Ans : B</b></p> | 1 | CO 2 | BT1 |
| 18 | <p>A recursive function without If and Else conditions will always lead to?</p> <p>A) Finite loop<br/>B) Infinite loop<br/>C) Incorrect result<br/>D) Correct result</p> <p><b>Ans : B</b></p>                             | 1 | CO 4 | BT2 |
| 19 | <p>Which variable has the longest scope?</p> <pre>#include &lt;stdio.h&gt; float a; int b; int main() {     int c;     return 0; }</pre> <p>A. a<br/>B. b<br/>C. c<br/>D. Both a and b</p> <p><b>Answer: d</b></p>         | 1 | CO 2 | BT1 |
| 20 | <p>Actual and formal parameters must agree in</p> <p>A. Data types<br/>B. Number of arguments and Data types<br/>C. Names and Data type<br/>D. None</p> <p style="text-align: right;">Answer: B</p>                        | 1 | CO 4 | BT2 |

**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (Autonomous)**  
**B.E. & B.Tech. I Sem (Main) Examination March 2021**

**Programming for Problem Solving**  
**(Common to all branches)**

**Time: 3 Hours****Max Marks: 60**

**Note:** Answer ALL questions from Part-A & Part-B (Internal Choice) at one place in the same order

**Part - A**  
**(5Q X 3M = 15 Marks)**

- |   |                                                                                 | M   | CO | BT |
|---|---------------------------------------------------------------------------------|-----|----|----|
| 1 | List the different components of a computer?                                    | (3) | 1  | L1 |
| 2 | What is recursion? Mention the limitations of recursion.                        | (3) | 4  | L1 |
| 3 | Give the syntax to declare 1-D arrays and explain how to access array elements. | (3) | 5  | L1 |
| 4 | What is the use of enumerated data types?                                       | (3) | 5  | L2 |
| 5 | Differentiate text file versus binary file.                                     | (3) | 6  | L4 |

**Part - B**  
**(5Q X 9M = 45 Marks)**

- |    |                                                                                                                                                                                                                 | M          | CO     | BT       |
|----|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|--------|----------|
| 6  | (a) List and explain the operators in 'C' with examples.<br>(b) Explain in detail about creating and running programs with a neat diagram.                                                                      | (5)<br>(4) | 3<br>1 | L1<br>L2 |
| 7  | (a) What is a Flowchart? Draw a flowchart to find the Sum of the digits of a given positive number.<br>(b) What is an expression? Explain about precedence and associativity of operators with simple examples. | (5)<br>(4) | 2<br>3 | L3<br>L2 |
| 8  | (a) Discuss the different looping statement in C with examples.<br>(b) Illustrate with examples different parameter passing mechanisms in C.                                                                    | (5)<br>(4) | 3<br>4 | L2<br>L2 |
| 9  | (a) Explain the different selective statements available in C.<br>(b) Define scope and lifetime of a variable. Explain using examples.                                                                          | (4)<br>(5) | 3<br>4 | L2<br>L2 |
| 10 | (a) Write a C program to implement binary search.<br>(b) Explain about any four string manipulation functions.                                                                                                  | (5)<br>(4) | 5<br>5 | L3<br>L2 |
| 11 | (a) Trace bubble sort technique for the following elements. {14, 33, 27, 10, 35, 19, 42, 44}. Show the array contents after each pass.<br>(b) Write a C function to simulate string copy operation.             | (5)<br>(4) | 5<br>4 | L3<br>L3 |

- 12 (a) Write a C program that uses a set of functions to perform the operations, reading, writing and multiply two complex numbers. Represent the complex number using a structure. (5) 5 L3

- (b) Give the syntax for declaring a pointer. Explain about pointer arithmetic using suitable examples. (4) 5 L2

**(OR)**

- 13 (a) Declare a structure to store the following information of an employee: Employee code, Employee name, Salary, Department number, Date of joining (it is itself a structure consisting of day, month and year). Write a C program to store the data of N employees and display the employee information who are drawing the maximum and minimum salary. (5) 5 L3

- (b) Compare and contrast structure and union with suitable examples. (4) 5 L4

- 14 (a) Explain different types of preprocessor directives with suitable examples. (5) 6 L2

- (b) Write a C program to display the contents of a file to standard output device. (4) 6 L3

**(OR)**

- 15 (a) Explain about error handling functions during file operations. (5) 6 L2

- (b) Write a C program to count the number of times a character occurs in a text file. (4) 6 L3

\*\*\*\*\*

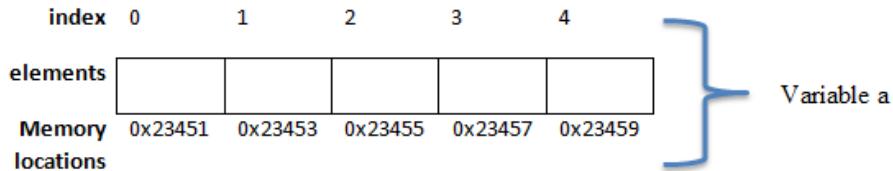
**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (Autonomous)**  
**B.E. & B.Tech. I Sem (Main) Examination March 2021**  
**Programming for Problem Solving Solution**  
**(Common to all branches)**

**Time: 3 Hours****Max Marks: 60**

**Note:** Answer ALL questions from **Part-A & Part -B (Internal Choice)** at one place in the same order

**Part - A**  
**(5Q X 3M = 15 Marks)**

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |     |
|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 1 | List the different components of a computer?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | (3) |
|   | There are 5 main computer components that are given below:<br>o Input Devices<br>o CPU<br>o Output Devices<br>o Primary Memory<br>o Secondary Memory                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |     |
| 2 | What is recursion? Mention the limitations of recursion.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | (3) |
|   | Recursion is a process in which the problem is specified in terms of itself. The function which calls itself is called as recursive function. A stopping condition must be specified to stop recursion else it will lead to an infinite process. Partial Solutions are combined to obtain the final solution.<br><br>Some of the Limitations are:<br><br>a. Execution time will be Slow and storing on the run-time stack more things is an important limitations of recursion.<br><br>b. If recursion is too deep, then there is a danger of running out of space on the stack and ultimately program crashes.<br><br>c. Cannot be applied to all problems . For simple problems the run time will take longer time.                                                                                                                                                  |     |
| 3 | Give the syntax to declare 1-D arrays and explain how to access array elements.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | (3) |
|   | <b>Declaration of one-dimensional or single-dimensional</b><br>Like any other variables, arrays must be declared before they are used. The general form of array declaration is<br><b>Syntax: datatype array_name[sizeofarray];</b><br>② The data type specifies the type of data/element that will be stored in the array, such as int, float, or char.<br>② The size indicates the maximum number of data/elements that can be stored in the array.<br>② The size of array should be a int constant or constant expression.<br>Let us take the same example to declare suitable one dimensional array variable to hold the data.<br><br>60 students marks secured in a subject<br><i>unsigned short int PPS[60];</i><br>• Salaries of 100 employees<br><i>float salaries_of_employees[100];</i><br><b>Internal representation /Memory Allocation</b><br><i>a[5];</i> |     |



4 What is the use of enumerated data types?

(3)

Enumeration is a user defined data type in C language. It is used to assign names to the integral constants which make a program easy to read and maintain. The keyword "enum" is used to declare an enumeration.

Here is the syntax of enum in C language,

Enum enum\_name{const1, const2, .....};

The enum keyword is also used to define the variables of enum type. There are two ways to define the

variables of enum type as follows.

enum week{sunday, monday, tuesday, wednesday, thursday, friday, saturday};

enum week day;

5 Differentiate text file versus binary file.

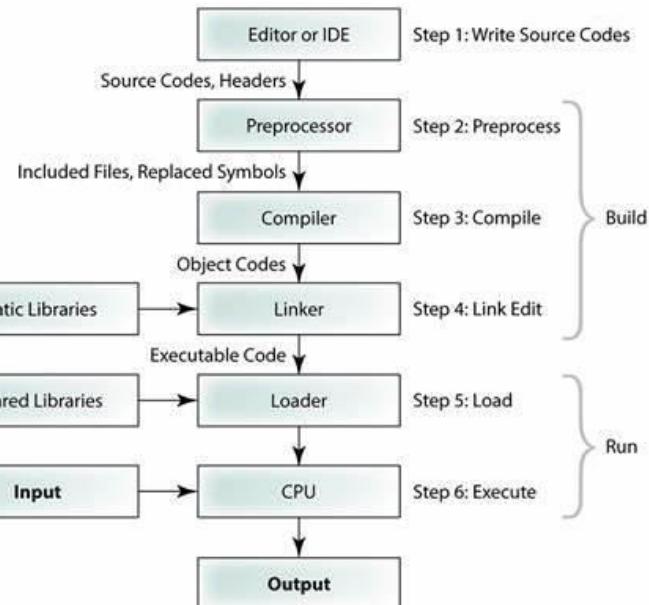
(3)

| Text File                                                                           | Binary File                                                                             |
|-------------------------------------------------------------------------------------|-----------------------------------------------------------------------------------------|
| Data is stored as lines of characters with each line terminated by newline.         | Data is stored on the disk in the same way as it is represented in the computer memory. |
| Human readable format.                                                              | Not in human readable format.                                                           |
| There is a special character called end-of-file(EOF) marker at the end of the file. | There is an end-of-file marker.                                                         |
| Data can be read using any of the text editors.                                     | Data can be read only by specific programs written for them.                            |

**Part - B**  
**(5Q X 9M = 45 Marks)**

|   |                                                                                                                                | M   |
|---|--------------------------------------------------------------------------------------------------------------------------------|-----|
| 6 | List and explain the operators in 'C' with examples.                                                                           | (5) |
|   | C programming language offers various types of operators having different functioning capabilities.<br>1. Arithmetic Operators |     |

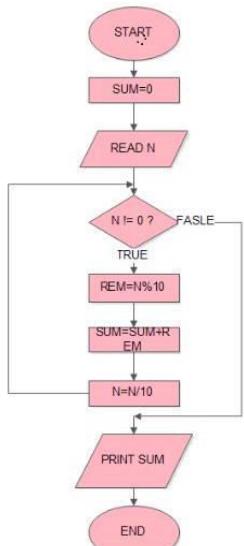
|  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |     |
|--|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|  | <p>2. Relational Operators<br/>     3. Logical Operators<br/>     4. Assignment Operators<br/>     5. Increment and Decrement Operators<br/>     6. Conditional Operator<br/>     7. Bitwise Operators<br/>     8. Special Operators</p> <p><b>Arithmetic Operators</b> are used to performing mathematical calculations like addition (+), subtraction (-), multiplication (*), division (/) and modulus (%).</p> <p><b>Relational operators</b> are used to comparing two quantities or values(&gt; ,&lt; ,&gt;= ,&lt;= ,== ,!= ).</p> <p><b>Logical Operators:</b>C provides three logical operators when we test more than one condition to make decisions. These are: &amp;&amp; (meaning logical AND),    (meaning logical OR) and ! (meaning logical NOT).</p> <p><b>Assignment operators</b> applied to assign the result of an expression to a variable. C has a collection of shorthand assignment operators.( =,+=,-=,*=,/=%=,&lt;&lt;=,&gt;=,&amp;=)</p> <p><b>Increment and Decrement Operators</b> are useful operators generally used to minimize the calculation, i.e. ++x and x++ means <math>x=x+1</math> or -x and x--means <math>x=x-1</math>. But there is a slight difference between ++ or -- written before or after the operand. Applying the pre-increment first add one to the operand and then the result is assigned to the variable on the left whereas post-increment first assigns the value to the variable on the left and then increment the operand.</p> <p>C offers a ternary operator which is the <b>conditional operator</b> (?: in combination) to construct conditional expressions.</p> <p><b>Bitwise Operators:</b>C provides a special operator for bit operation between two variables(&lt;&lt;,&gt;&gt;,~,^, )</p> <p>C supports some <b>special operators</b>(sizeof(),&amp;.*,-&gt;)</p> <p><b>Note:</b> Any three operators examples are considered.</p> |     |
|  | Explain in detail about creating and running programs with a neat diagram.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | (4) |



(OR)

7 What is a Flowchart? Draw a flowchart to find the Sum of the digits of a given positive number. (5)

**Flowchart** is a diagrammatic representation of sequence of logical steps of a program.



What is an expression? Explain about precedence and associativity of operators with simple examples. (4)

The precedence of operators comes into effect whenever there are more than one operators involved in the expression to be evaluated.

- ② There are many levels of precedence, each containing one or more operators.
- ② The operator with the highest precedence level is evaluated first.
- ② Operators in the same level are evaluated either from left to right or right to left depending on their associativity. Associativity determines the operations to be performed from right and from left, when several operators of the same priority level exist in the same expression.
- ② The order of precedence and associativity of operators are two very important aspects

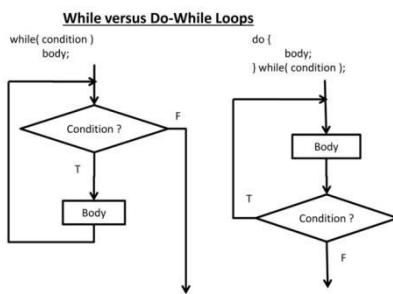
|                                   | <p>of evaluation.</p> <table border="1"> <thead> <tr> <th>OPERATOR</th><th>TYPE</th><th>ASSOCIATIVITY</th></tr> </thead> <tbody> <tr><td>( ) [ ] . -&gt;</td><td></td><td>left-to-right</td></tr> <tr><td>++ -- + - ! ~ (type) * &amp; sizeof</td><td>Unary Operator</td><td>right-to-left</td></tr> <tr><td>* / %</td><td>Arithmetic Operator</td><td>left-to-right</td></tr> <tr><td>+ -</td><td>Arithmetic Operator</td><td>left-to-right</td></tr> <tr><td>&lt;&lt; &gt;&gt;</td><td>Shift Operator</td><td>left-to-right</td></tr> <tr><td>&lt; &lt;= &gt; &gt;=</td><td>Relational Operator</td><td>left-to-right</td></tr> <tr><td>== !=</td><td>Relational Operator</td><td>left-to-right</td></tr> <tr><td>&amp;</td><td>Bitwise AND Operator</td><td>left-to-right</td></tr> <tr><td>^</td><td>Bitwise EX-OR Operator</td><td>left-to-right</td></tr> <tr><td> </td><td>Bitwise OR Operator</td><td>left-to-right</td></tr> <tr><td>&amp;&amp;</td><td>Logical AND Operator</td><td>left-to-right</td></tr> <tr><td>  </td><td>Logical OR Operator</td><td>left-to-right</td></tr> <tr><td>? :</td><td>Ternary Conditional Operator</td><td>right-to-left</td></tr> <tr><td>= += -= *= /= %= &amp;= ^=  = &lt;&lt;= &gt;&gt;=</td><td>Assignment Operator</td><td>right-to-left</td></tr> <tr><td>,</td><td>Comma</td><td>left-to-right</td></tr> </tbody> </table>                                                      | OPERATOR      | TYPE | ASSOCIATIVITY | ( ) [ ] . -> |  | left-to-right | ++ -- + - ! ~ (type) * & sizeof | Unary Operator | right-to-left | * / % | Arithmetic Operator | left-to-right | + - | Arithmetic Operator | left-to-right | << >> | Shift Operator | left-to-right | < <= > >= | Relational Operator | left-to-right | == != | Relational Operator | left-to-right | & | Bitwise AND Operator | left-to-right | ^ | Bitwise EX-OR Operator | left-to-right |  | Bitwise OR Operator | left-to-right | && | Logical AND Operator | left-to-right |  | Logical OR Operator | left-to-right | ? : | Ternary Conditional Operator | right-to-left | = += -= *= /= %= &= ^=  = <<= >>= | Assignment Operator | right-to-left | , | Comma | left-to-right |  |
|-----------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------|------|---------------|--------------|--|---------------|---------------------------------|----------------|---------------|-------|---------------------|---------------|-----|---------------------|---------------|-------|----------------|---------------|-----------|---------------------|---------------|-------|---------------------|---------------|---|----------------------|---------------|---|------------------------|---------------|--|---------------------|---------------|----|----------------------|---------------|--|---------------------|---------------|-----|------------------------------|---------------|-----------------------------------|---------------------|---------------|---|-------|---------------|--|
| OPERATOR                          | TYPE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | ASSOCIATIVITY |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| ( ) [ ] . ->                      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | left-to-right |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| ++ -- + - ! ~ (type) * & sizeof   | Unary Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | right-to-left |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| * / %                             | Arithmetic Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | left-to-right |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| + -                               | Arithmetic Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | left-to-right |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| << >>                             | Shift Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | left-to-right |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| < <= > >=                         | Relational Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | left-to-right |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| == !=                             | Relational Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | left-to-right |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| &                                 | Bitwise AND Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | left-to-right |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| ^                                 | Bitwise EX-OR Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | left-to-right |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
|                                   | Bitwise OR Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | left-to-right |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| &&                                | Logical AND Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | left-to-right |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
|                                   | Logical OR Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | left-to-right |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| ? :                               | Ternary Conditional Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | right-to-left |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| = += -= *= /= %= &= ^=  = <<= >>= | Assignment Operator                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | right-to-left |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| ,                                 | Comma                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | left-to-right |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
| 8                                 | <p>Discuss the different looping statement in C with examples.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | (5)           |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |
|                                   | <p>Computers are very good at performing repetitive tasks very quickly. Computer repeat actions either a specified number of times or until some stopping condition is met.</p> <p>while Loops ( Condition-Controlled Loops )</p> <ul style="list-style-type: none"> <li>Both while loops and do-while loops ( see below ) are condition-controlled, meaning that they continue to loop until some condition is met.</li> <li>Both while and do-while loops alternate between performing actions and testing for the stopping condition.</li> <li>While loops check for the stopping condition first, and may not execute the body of the loop at all if the condition is initially false.</li> <li>Syntax:</li> </ul> <pre>while( condition )     body;</pre> <p>where the body can be either a single statement or a block of statements within { curly braces }.</p> <ul style="list-style-type: none"> <li>Example:</li> </ul> <pre>int i = 0; while( i &lt; 5 )     printf( "i = %d\n", i++ ); printf( "After loop, i = %d\n", i );</pre> <p>do-while Loops</p> <ul style="list-style-type: none"> <li>do-while loops are exactly like while loops, except that the test is performed at the end of the loop rather than the beginning.</li> <li>This guarantees that the loop will be performed at least once, which is useful for checking user input among other things ( see example below. )</li> <li>Syntax:</li> </ul> |               |      |               |              |  |               |                                 |                |               |       |                     |               |     |                     |               |       |                |               |           |                     |               |       |                     |               |   |                      |               |   |                        |               |  |                     |               |    |                      |               |  |                     |               |     |                              |               |                                   |                     |               |   |       |               |  |

```
do {
    body;
} while( condition );
```

- In theory the body can be either a single statement or a block of statements within { curly braces }, but in practice the curly braces are almost always used with do-whiles.
- Example:

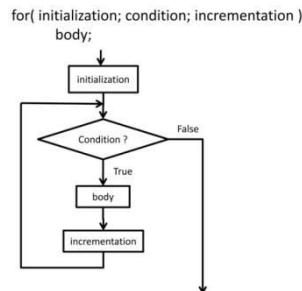
```
int month;
do {
    printf( "Please enter the month of your birth > " );
    scanf( "%d", &month );
} while ( month < 1 || month > 12 );
```

- Note that the above example could be improved by reporting to the user what the problem is if month is not in the range 1 to 12, and that it could also be done using a while loop if month were initialized to a value that ensures entering the loop.
- The following diagram shows the difference between while and do-while loops. Note that once you enter the loop, the operation is identical from that point forward:



### for Loops

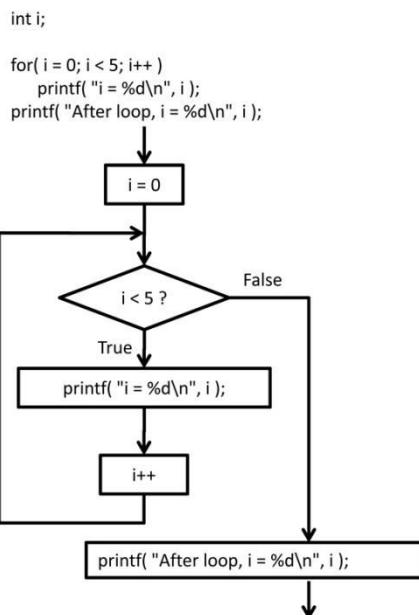
- for-loops are counter-controlled, meaning that they are normally used whenever the number of iterations is known in advance.
- Syntax:



where again the body can be either a single statement or a block of statements within { curly braces }.

- Details:

- The initialization step occurs one time only, before the loop begins.
- The condition is tested at the beginning of each iteration of the loop.
  - If the condition is true ( non-zero ), then the body of the loop is executed next.
  - If the condition is false ( zero ), then the body is not executed, and execution continues with the code following the loop.
- The incrementation happens AFTER the execution of the body, and only when the body is executed.
- Example:



Illustrate with examples different parameter passing mechanisms in C.

(4)

There are two parameter passing mechanisms

1. call by value. Or passing by value
2. call by reference or passing by address or call by address.

#### **Call by value:**

Passing arguments by value means , the contents of the arguments in the calling function are not changed , even if they are changed in the called function.

This is because the content of the variable is copied to the formal parameter of the function definition, thus preventing the contents of the argument in the calling function.

#### **Call by reference:**

Call by reference means sending the address of variables as arguments to the function. When addresses are sent, the changes occurred in the called function can also effect in the calling function.

*example to illustrate call by value and call by reference\*/*

```
#include <stdio.h>
void main()
{
void makezero(int,int*); /*function prototype*/
int x=50,y=100;
makezero(x,&y); /* function call by value and referene */
```

|   |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |     |
|---|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|   | <pre> printf("x=%d y=%d", x,y); } /*function makezero */ void makezero(int a,int *b) { a=0; *b=0; } <b>OUTPUT</b> x=50 y=0 </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |     |
|   | <b>(OR)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |     |
| 9 | Explain the different selective statements available in C.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   | (4) |
|   | <p>Control Structures - Intro, Selection</p> <p>Flow of Control:</p> <p>Flow of control through any given function is implemented with three basic types of control structures:</p> <ul style="list-style-type: none"> <li>• Sequential: default mode. Sequential execution of code statements (one line after another) -- like following a recipe</li> <li>• Selection: used for decisions, branching -- choosing between 2 or more alternative paths. In C++, these are the types of selection statements: <ul style="list-style-type: none"> <li>◦ if</li> <li>◦ if/else</li> <li>◦ switch</li> </ul> </li> <li>• Repetition: used for looping, i.e. repeating a piece of code multiple times in a row. In C++, there are three types of loops: <ul style="list-style-type: none"> <li>• while</li> <li>• do/while</li> <li>• for</li> </ul> </li> </ul> <p>The function construct, itself, forms another way to affect flow of control through a whole program.</p> <p>f-else</p> <ul style="list-style-type: none"> <li>• "if" blocks and "if-else" blocks are used to make decisions, and to optionally execute certain code.</li> <li>• The general syntax of the if-else structure is as follows: <ul style="list-style-type: none"> <li>• if( condition )</li> <li>•     true_block</li> <li>•     else</li> <li>•         false_block</li> </ul> </li> <li>• Either the true_block or the false_block can be either a single statement or a block of statements enclosed in { curly braces }</li> <li>• The else clause and the false block are optional.</li> <li>• In execution, the condition is evaluated for truth or falsehood. <ul style="list-style-type: none"> <li>◦ If the condition evaluates to true ( non-zero ), then the true_block of code</li> </ul> </li> </ul> |     |

|    |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         |     |
|----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|    | <p>is executed.</p> <ul style="list-style-type: none"> <li>○ If the condition evaluates to false ( zero ), then the false-block of code is executed if it is present.</li> <li>○ After one or the other block is executed, then execution continues with whatever code follows the if-else construct, without executing the other block.</li> </ul> <ul style="list-style-type: none"> <li>• Example:</li> <li>•   if( x &lt; 0.0 ) {</li> <li>•       printf( "Error - The x coordinate cannot be negative. ( x = %f ) \n", x );</li> <li>•       exit( -1 );</li> <li>• }</li> </ul> <p><b>Switch statement in C</b> tests the value of a variable and compares it with multiple cases. Once the case match is found, a block of statements associated with that particular case is executed.</p> <pre>switch( expression ) {     case value-1:         Block-1;         Break;     case value-2:         Block-2;         Break;     case value-n:         Block-n;         Break;     default:         Block-1;         Break; } Statement-x;</pre> |     |
|    | Define scope and lifetime of a variable. Explain using examples.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        | (5) |
| 10 | Write a C program to implement binary search.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                           | (5) |
|    | Program to implement Binary search */ <pre>#include&lt;stdio.h&gt; int main() { int n; printf("\n Enter the size of the array:"); scanf("%d",&amp;n); int a[n],i,key,FOUND=0; printf("\n Enter elements in increasing order"); for(i=0;i&lt;n;i++) scanf("%d",&amp;a[i]); printf("\n Enter element for searching:"); scanf("%d",&amp;key); int low=0,high=n-1,mid; while(low&lt;=high) {</pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |     |

```

mid=(low+high)/2;
if(key>a[mid])
low=mid+1;
else if(key<a[mid])
high=mid-1;
else
{
printf("\n Found at %d",i);
FOUND=1; break;
}
}
if(FOUND == 0)
printf("\n NOT FOUND...");
return 0;
}

```

Explain about any four string manipulation functions.

(4)

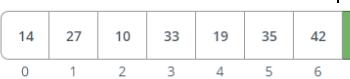
#### **String Manipulation:**

The string header, string.h, provides many functions useful for manipulating strings or character arrays. Some of these are mentioned below:

| Function                      | Description                                                                                                                                                                                                                                                                                                      |
|-------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>strcpy(s1,s2)</code>    | Copies s2 into s1                                                                                                                                                                                                                                                                                                |
| <code>strcat(s1,s2)</code>    | Concatenates s2 to s1. That is, it appends the string contained by s2 to the end of the string pointed to by s1. The terminating null character of s1 is overwritten. Copying stops once the terminating null character of s2 is copied.                                                                         |
| <code>strncat(s1,s2,n)</code> | Appends the string pointed to by s2 to the end of the string pointed to by s1 up to n characters long. The terminating null character of s1 is overwritten. Copying stops once n characters are copied or the terminating null character of s2 is copied. A terminating null character is always appended to s1. |
| <code>strlen(s1)</code>       | Returns the length of s1. That is, it returns the number of characters in the string without the terminating null character.                                                                                                                                                                                     |
| <code>strcmp(s1,s2)</code>    | Returns 0 if s1 and s2 are the same<br>Returns less than 0 if s1<s2<br>Returns greater than 0 if s1>s2                                                                                                                                                                                                           |
| <code>strchr(s1,ch)</code>    | Returns pointer to first occurrence ch in s1                                                                                                                                                                                                                                                                     |
| <code>strstr(s1,s2)</code>    | Returns pointer to first occurrence s2 in s1                                                                                                                                                                                                                                                                     |

(OR)

|    |                                                                                                                                       |                                                                                                                                                                                                                                                                        |    |    |    |    |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
|----|---------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----|----|----|----|----|--|----|----|-----|----|----|----|----|----|---|---|---|---|---|---|----|---|---|--|--|
| 11 | Trace bubble sort technique for the following elements.<br>{14, 33, 27, 10, 35, 19, 42, 44}. Show the array contents after each pass. |                                                                                                                                                                                                                                                                        |    |    |    |    |    |  |    |    | (5) |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
|    | 1                                                                                                                                     | <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>14</td><td>33</td><td>27</td><td>10</td><td>35</td><td>19</td><td>42</td><td>44</td></tr> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td></tr> </table> |    |    |    |    |    |  | 14 | 33 | 27  | 10 | 35 | 19 | 42 | 44 | 0 | 1 | 2 | 3 | 4 | 5 | 6  | 7 | 2 |  |  |
| 14 | 33                                                                                                                                    | 27                                                                                                                                                                                                                                                                     | 10 | 35 | 19 | 42 | 44 |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
| 0  | 1                                                                                                                                     | 2                                                                                                                                                                                                                                                                      | 3  | 4  | 5  | 6  | 7  |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
|    | 3                                                                                                                                     | <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>14</td><td>33</td><td>27</td><td>10</td><td>35</td><td>19</td><td>42</td></tr> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table>                      |    |    |    |    |    |  | 14 | 33 | 27  | 10 | 35 | 19 | 42 | 0  | 1 | 2 | 3 | 4 | 5 | 6 | 4  |   |   |  |  |
| 14 | 33                                                                                                                                    | 27                                                                                                                                                                                                                                                                     | 10 | 35 | 19 | 42 |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
| 0  | 1                                                                                                                                     | 2                                                                                                                                                                                                                                                                      | 3  | 4  | 5  | 6  |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
|    | 5                                                                                                                                     | <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>14</td><td>33</td><td>27</td><td>10</td><td>35</td><td>19</td><td>42</td></tr> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table>                      |    |    |    |    |    |  | 14 | 33 | 27  | 10 | 35 | 19 | 42 | 0  | 1 | 2 | 3 | 4 | 5 | 6 | 6  |   |   |  |  |
| 14 | 33                                                                                                                                    | 27                                                                                                                                                                                                                                                                     | 10 | 35 | 19 | 42 |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
| 0  | 1                                                                                                                                     | 2                                                                                                                                                                                                                                                                      | 3  | 4  | 5  | 6  |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
|    | 7                                                                                                                                     | <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>14</td><td>27</td><td>33</td><td>10</td><td>35</td><td>19</td><td>42</td></tr> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table>                      |    |    |    |    |    |  | 14 | 27 | 33  | 10 | 35 | 19 | 42 | 0  | 1 | 2 | 3 | 4 | 5 | 6 | 8  |   |   |  |  |
| 14 | 27                                                                                                                                    | 33                                                                                                                                                                                                                                                                     | 10 | 35 | 19 | 42 |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
| 0  | 1                                                                                                                                     | 2                                                                                                                                                                                                                                                                      | 3  | 4  | 5  | 6  |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
|    | 9                                                                                                                                     | <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>14</td><td>27</td><td>10</td><td>33</td><td>35</td><td>19</td><td>42</td></tr> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table>                      |    |    |    |    |    |  | 14 | 27 | 10  | 33 | 35 | 19 | 42 | 0  | 1 | 2 | 3 | 4 | 5 | 6 | 10 |   |   |  |  |
| 14 | 27                                                                                                                                    | 10                                                                                                                                                                                                                                                                     | 33 | 35 | 19 | 42 |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
| 0  | 1                                                                                                                                     | 2                                                                                                                                                                                                                                                                      | 3  | 4  | 5  | 6  |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
|    | 11                                                                                                                                    | <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>14</td><td>27</td><td>10</td><td>33</td><td>35</td><td>19</td><td>42</td></tr> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table>                      |    |    |    |    |    |  | 14 | 27 | 10  | 33 | 35 | 19 | 42 | 0  | 1 | 2 | 3 | 4 | 5 | 6 | 12 |   |   |  |  |
| 14 | 27                                                                                                                                    | 10                                                                                                                                                                                                                                                                     | 33 | 35 | 19 | 42 |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
| 0  | 1                                                                                                                                     | 2                                                                                                                                                                                                                                                                      | 3  | 4  | 5  | 6  |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
|    | 13                                                                                                                                    | <table border="1" style="margin-left: auto; margin-right: auto;"> <tr><td>14</td><td>27</td><td>10</td><td>33</td><td>19</td><td>35</td><td>42</td></tr> <tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr> </table>                      |    |    |    |    |    |  | 14 | 27 | 10  | 33 | 19 | 35 | 42 | 0  | 1 | 2 | 3 | 4 | 5 | 6 | 14 |   |   |  |  |
| 14 | 27                                                                                                                                    | 10                                                                                                                                                                                                                                                                     | 33 | 19 | 35 | 42 |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |
| 0  | 1                                                                                                                                     | 2                                                                                                                                                                                                                                                                      | 3  | 4  | 5  | 6  |    |  |    |    |     |    |    |    |    |    |   |   |   |   |   |   |    |   |   |  |  |

|  |                                                                                                                                                                                                                                                                                                                                                                                      |                                                                                                                                                                              |                           |  |
|--|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------------------------|--|
|  | 15                                                                                                                                                                                                                                                                                                                                                                                   |  <p>Steps:<br/>Done this pass. The last element processed is now in its final position.</p> | So on with other elements |  |
|  | Write a C function to simulate string copy operation.                                                                                                                                                                                                                                                                                                                                |                                                                                                                                                                              |                           |  |
|  | <pre>#include &lt;stdio.h&gt; void copy_string(char [], char []);  int main() {     char s[50], d[50];      printf("Input a string\n");     gets(s);      copy_string(d, s);     printf("The string: %s\n", d);      return 0; }  void copy_string(char d[], char s[]) {     int c = 0;      while (s[c] != '\0') {         d[c] = s[c];         c++;     }     d[c] = '\0'; }</pre> |                                                                                                                                                                              |                           |  |

|    |     |                                                                                                                                                                                       |     |
|----|-----|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 12 | (a) | Write a C program that uses a set of functions to perform the operations, reading, writing and multiply two complex numbers. Represent the complex number using a structure.          | (5) |
|    |     | <pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; struct complex {     int real, img; };  int main() {     int choice, x, y, z;     struct complex a, b, c;      while(1)</pre> |     |

```

{
printf("Press 1 to add two complex numbers.\n");
printf("Press 2 to subtract two complex numbers.\n");
printf("Press 3 to multiply two complex numbers.\n");
printf("Press 4 to divide two complex numbers.\n");
printf("Press 5 to exit.\n");
printf("Enter your choice\n");
scanf("%d", &choice);

if (choice == 5)
    exit(0);

if (choice >= 1 && choice <= 4)
{
    printf("Enter a and b where a + ib is the first complex number.");
    printf("\na = ");
    scanf("%d", &a.real);
    printf("b = ");
    scanf("%d", &a.img);
    printf("Enter c and d where c + id is the second complex number.");
    printf("\nc = ");
    scanf("%d", &b.real);
    printf("d = ");
    scanf("%d", &b.img);
}
if (choice == 1)
{
    c.real = a.real + b.real;
    c.img = a.img + b.img;

    if (c.img >= 0)
        printf("Sum of the complex numbers = %d + %di", c.real, c.img);
    else
        printf("Sum of the complex numbers = %d %di", c.real, c.img);
}
else if (choice == 2)
{
    c.real = a.real - b.real;
    c.img = a.img - b.img;

    if (c.img >= 0)
        printf("Difference of the complex numbers = %d + %di", c.real, c.img);
    else
        printf("Difference of the complex numbers = %d %di", c.real, c.img);
}
else if (choice == 3)
{
    c.real = a.real*b.real - a.img*b.img;
    c.img = a.img*b.real + a.real*b.img;

    if (c.img >= 0)
        printf("Multiplication of the complex numbers = %d + %di", c.real, c.img);
    else
        printf("Multiplication of the complex numbers = %d %di", c.real, c.img);
}

```

```

}
else if (choice == 4)
{
    if (b.real == 0 && b.img == 0)
        printf("Division by 0 + 0i isn't allowed.");
    else
    {
        x = a.real*b.real + a.img*b.img;
        y = a.img*b.real - a.real*b.img;
        z = b.real*b.real + b.img*b.img;

        if (x%z == 0 && y%z == 0)
        {
            if (y/z >= 0)
                printf("Division of the complex numbers = %d + %di", x/z, y/z);
            else
                printf("Division of the complex numbers = %d %di", x/z, y/z);
        }
        else if (x%z == 0 && y%z != 0)
        {
            if (y/z >= 0)
                printf("Division of two complex numbers = %d + %d/%di", x/z, y, z);
            else
                printf("Division of two complex numbers = %d %d/%di", x/z, y, z);
        }
        else if (x%z != 0 && y%z == 0)
        {
            if (y/z >= 0)
                printf("Division of two complex numbers = %d/%d + %di", x, z, y/z);
            else
                printf("Division of two complex numbers = %d %d/%di", x, z, y/z);
        }
        else
        {
            if (y/z >= 0)
                printf("Division of two complex numbers = %d/%d + %d/%di", x, z, y, z);
            else
                printf("Division of two complex numbers = %d %d/%d/%di", x, z, y, z);
        }
    }
}
else
    printf("Invalid choice.");

    printf("\nPress any key to enter choice again...\n");
}
}

```

|  |                                                                                                                                                                                                                                                    |     |
|--|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|  | (b) Give the syntax for declaring a pointer. Explain about pointer arithmetic using suitable examples.                                                                                                                                             | (4) |
|  | <p><b>Declaring a Pointer – Pointer Declaration :</b></p> <ul style="list-style-type: none"> <li>• Like other variables, in a program, a pointer has to be declared.</li> <li>• A pointer will have a value, scope, lifetime, and name.</li> </ul> |     |

|        |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |     |
|--------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|        | <ul style="list-style-type: none"> <li>• Pointers will occupy a certain number of memory locations.</li> <li>• The pointer operator available in C is ‘*’ , called ‘value at address’ operator.</li> <li>• The syntax for declaring a pointer variable is<br/><b>datatype * pointer_variable</b></li> <li>• The data type of pointer and the variable must match, an int pointer can hold the address of int variable, similarly a pointer declared with float data type can hold the address of a float variable.</li> </ul> <p>The operations are:</p> <ol style="list-style-type: none"> <li>1. Increment/Decrement of a Pointer</li> <li>2. Addition of integer to a pointer</li> <li>3. Subtraction of integer to a pointer</li> <li>4. Subtracting two pointers of the same type</li> </ol> |     |
|        | <b>(OR)</b>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |     |
| 13 (a) | <p>Declare a structure to store the following information of an employee: Employee code, Employee name, Salary, Department number, Date of joining (it is itself a structure consisting of day, month and year).</p> <p>Write a C program to store the data of N employees and display the employee information who are drawing the maximum and minimum salary.</p>                                                                                                                                                                                                                                                                                                                                                                                                                               | (5) |

|                           | <pre> printf("-----\n"); } high=emp[0].salary; for(i=0;i&lt;n;i++) {     if(emp[i].salary&gt;high)         high=emp[i].salary; } printf("Highest          salary          employee      details:"); printf("\n-----\n"); printf("EMPNO  NAME SALARY\n"); for(i=0;i&lt;n;i++) {     if(emp[i].salary==high)         printf("\n      %d\t%s\t%d",emp[i].eno,emp[i].ename,emp[i].salary); } low=emp[0].salary; for(i=0;i&lt;n;i++) {     if(emp[i].salary&lt;low)         low=emp[i].salary; } printf("lowest          salary          employee      details:"); printf("\n-----\n"); printf("EMPNO  NAME SALARY\n"); for(i=0;i&lt;n;i++) {     if(emp[i].salary==low)         printf("\n      %d\t%s\t%d",emp[i].eno,emp[i].ename,emp[i].salary); } return 0; } </pre>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |                                                                                                                                                                                        |           |       |         |                                                               |                                                           |      |                                                                                                                                                                                       |                                                                                                                                                                                        |        |                                                                             |                                                            |                |                                                                                |                                                                         |                   |                                              |                                            |                           |                                                        |                                                      |  |
|---------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------|-------|---------|---------------------------------------------------------------|-----------------------------------------------------------|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|-----------------------------------------------------------------------------|------------------------------------------------------------|----------------|--------------------------------------------------------------------------------|-------------------------------------------------------------------------|-------------------|----------------------------------------------|--------------------------------------------|---------------------------|--------------------------------------------------------|------------------------------------------------------|--|
|                           | (b) Compare and contrast structure and union with suitable examples.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | (4)                                                                                                                                                                                    |           |       |         |                                                               |                                                           |      |                                                                                                                                                                                       |                                                                                                                                                                                        |        |                                                                             |                                                            |                |                                                                                |                                                                         |                   |                                              |                                            |                           |                                                        |                                                      |  |
|                           | <table border="1"> <thead> <tr> <th></th><th>STRUCTURE</th><th>UNION</th></tr> </thead> <tbody> <tr> <td>Keyword</td><td>The keyword <code>struct</code> is used to define a structure</td><td>The keyword <code>union</code> is used to define a union.</td></tr> <tr> <td>Size</td><td>When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is greater than or equal to the sum of sizes of its members.</td><td>When a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of union is equal to the size of largest member.</td></tr> <tr> <td>Memory</td><td>Each member within a structure is assigned unique storage area or location.</td><td>Memory allocated is shared by individual members of union.</td></tr> <tr> <td>Value Altering</td><td>Altering the value of a member will not affect other members of the structure.</td><td>Altering the value of any of the member will alter other member values.</td></tr> <tr> <td>Accessing members</td><td>Individual member can be accessed at a time.</td><td>Only one member can be accessed at a time.</td></tr> <tr> <td>Initialization of Members</td><td>Several members of a structure can initialize at once.</td><td>Only the first member of a union can be initialized.</td></tr> </tbody> </table> |                                                                                                                                                                                        | STRUCTURE | UNION | Keyword | The keyword <code>struct</code> is used to define a structure | The keyword <code>union</code> is used to define a union. | Size | When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is greater than or equal to the sum of sizes of its members. | When a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of union is equal to the size of largest member. | Memory | Each member within a structure is assigned unique storage area or location. | Memory allocated is shared by individual members of union. | Value Altering | Altering the value of a member will not affect other members of the structure. | Altering the value of any of the member will alter other member values. | Accessing members | Individual member can be accessed at a time. | Only one member can be accessed at a time. | Initialization of Members | Several members of a structure can initialize at once. | Only the first member of a union can be initialized. |  |
|                           | STRUCTURE                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | UNION                                                                                                                                                                                  |           |       |         |                                                               |                                                           |      |                                                                                                                                                                                       |                                                                                                                                                                                        |        |                                                                             |                                                            |                |                                                                                |                                                                         |                   |                                              |                                            |                           |                                                        |                                                      |  |
| Keyword                   | The keyword <code>struct</code> is used to define a structure                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | The keyword <code>union</code> is used to define a union.                                                                                                                              |           |       |         |                                                               |                                                           |      |                                                                                                                                                                                       |                                                                                                                                                                                        |        |                                                                             |                                                            |                |                                                                                |                                                                         |                   |                                              |                                            |                           |                                                        |                                                      |  |
| Size                      | When a variable is associated with a structure, the compiler allocates the memory for each member. The size of structure is greater than or equal to the sum of sizes of its members.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | When a variable is associated with a union, the compiler allocates the memory by considering the size of the largest memory. So, size of union is equal to the size of largest member. |           |       |         |                                                               |                                                           |      |                                                                                                                                                                                       |                                                                                                                                                                                        |        |                                                                             |                                                            |                |                                                                                |                                                                         |                   |                                              |                                            |                           |                                                        |                                                      |  |
| Memory                    | Each member within a structure is assigned unique storage area or location.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | Memory allocated is shared by individual members of union.                                                                                                                             |           |       |         |                                                               |                                                           |      |                                                                                                                                                                                       |                                                                                                                                                                                        |        |                                                                             |                                                            |                |                                                                                |                                                                         |                   |                                              |                                            |                           |                                                        |                                                      |  |
| Value Altering            | Altering the value of a member will not affect other members of the structure.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | Altering the value of any of the member will alter other member values.                                                                                                                |           |       |         |                                                               |                                                           |      |                                                                                                                                                                                       |                                                                                                                                                                                        |        |                                                                             |                                                            |                |                                                                                |                                                                         |                   |                                              |                                            |                           |                                                        |                                                      |  |
| Accessing members         | Individual member can be accessed at a time.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | Only one member can be accessed at a time.                                                                                                                                             |           |       |         |                                                               |                                                           |      |                                                                                                                                                                                       |                                                                                                                                                                                        |        |                                                                             |                                                            |                |                                                                                |                                                                         |                   |                                              |                                            |                           |                                                        |                                                      |  |
| Initialization of Members | Several members of a structure can initialize at once.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    | Only the first member of a union can be initialized.                                                                                                                                   |           |       |         |                                                               |                                                           |      |                                                                                                                                                                                       |                                                                                                                                                                                        |        |                                                                             |                                                            |                |                                                                                |                                                                         |                   |                                              |                                            |                           |                                                        |                                                      |  |
| 14                        | (a) Explain different types of preprocessor directives with suitable examples.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            | (5)                                                                                                                                                                                    |           |       |         |                                                               |                                                           |      |                                                                                                                                                                                       |                                                                                                                                                                                        |        |                                                                             |                                                            |                |                                                                                |                                                                         |                   |                                              |                                            |                           |                                                        |                                                      |  |
|                           | <p style="text-align: center;"><b>Syntax/Description</b></p> <p><b>Preprocessor</b></p> <p><b>Syntax:</b> <code>#define</code><br/>This macro defines constant value and can be any of the basic data types.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          |                                                                                                                                                                                        |           |       |         |                                                               |                                                           |      |                                                                                                                                                                                       |                                                                                                                                                                                        |        |                                                                             |                                                            |                |                                                                                |                                                                         |                   |                                              |                                            |                           |                                                        |                                                      |  |

|     |  |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |     |
|-----|--|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
|     |  | <p>Header file inclusion</p> <p><b>Syntax:</b> #include &lt;file_name&gt;<br/>The source code of the file "file_name" is included in the main program at the specified place.</p> <p>Conditional compilation</p> <p><b>Syntax:</b> #ifdef, #endif, #if, #else, #ifndef<br/>Set of commands are included or excluded in source program before compilation with respect to the condition.</p> <p>Other directives</p> <p><b>Syntax:</b> #undef, #pragma<br/>#undef is used to undefine a defined macro variable. #Pragma is used to call a function before and after main function in a C program.</p> |     |
| (b) |  | Write a C program to display the contents of a file to standard output device.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | (4) |
|     |  | <pre>#include &lt;stdio.h&gt; #include &lt;stdlib.h&gt; // For exit()  int main() {     FILE *fptr;      char filename[100], c;      printf("Enter the filename to open \n");     scanf("%s", filename);      // Open file     fptr = fopen(filename, "r");     if (fptr == NULL)     {         printf("Cannot open file \n");         exit(0);     }      // Read contents from file     c = fgetc(fptr);     while (c != EOF)     {         printf ("%c", c);         c = fgetc(fptr);     }      fclose(fptr);     return 0; }</pre>                                                              |     |

(OR)

|    |     |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                            |     |
|----|-----|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 15 | (a) | Explain about error handling functions during file operations.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | (5) |
|    |     | <p>C language uses the following functions to represent error messages associated with <b>errno</b>:</p> <ul style="list-style-type: none"> <li>• perror(): returns the string passed to it along with the textual representation of the current errno value.</li> <li>• strerror() is defined in <b>string.h</b> library. This method returns a pointer to the string representation of the current errno value.</li> </ul> <p>Other ways of Error Handling</p> <p>We can also use <b>Exit Status</b> constants in the exit() function to inform the calling function about the error. The two constant values available for use are EXIT_SUCCESS and EXIT_FAILURE. These are nothing but macros defined <b>stdlib.h</b> header file.</p> |     |
|    | (b) | Write a C program to count the number of times a character occurs in a text file.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | (4) |
|    |     | <pre>#include&lt;conio.h&gt; void main() {     char ch;     int count=0;     FILE *fptr;     clrscr();     fptr=fopen("text.txt","w");     if(fptr==NULL) {         printf("File can't be created\a");         getch();         exit(0);     }     printf("Enter some text and press enter key:\n");     while((ch=getche())!='\r') {         fputc(ch,fptr);     }     fclose(fptr);     fptr=fopen("text.txt","r");     printf("\nContents of the File is:");     while((ch=fgetc(fptr))!=EOF) {         count++;         printf("%c",ch);     }     fclose(fptr);     printf("\nThe number of characters present in file is: %d",count);     getch(); }</pre>                                                                           |     |

**R 20**

**Code No.: 20CSC01**

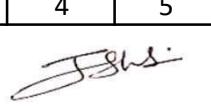
**CHAITANYA BHARATHI INSTITUTE OF TECHNOLOGY (A)**  
**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**First Year A.Y. 2020-21 CIE**

Name of the Faculty: Sri J Shiva Sai

Branch: Biotech(K)

| SNo | HallticketNo | Name                | CT-1(20) | CT-2(20) | AVG(20) | ASSIG-1 | ASSIG-2 | CS(10) | AVG(10) | S T-1(5) | S T-2(5) | S T-3(5) | AVG(5) | ATT (5) | TOT(40) |
|-----|--------------|---------------------|----------|----------|---------|---------|---------|--------|---------|----------|----------|----------|--------|---------|---------|
| 1   | 160120805001 | ADITHI REDDI K      | 17       | 2        | 10      | 10      | 8       | 6      | 9       | 3        | 5        | 3        | 4      | 5       | 28      |
| 2   | 160120805002 | AISHWARYA K         | 16       | 10       | 13      | 10      | 8       | -      | 9       | 4        | 4        | 4        | 4      | 5       | 31      |
| 3   | 160120805003 | ALWINA G            | 12       | 4        | 8       | 10      | 9       | -      | 10      | 2        | 4        | 4        | 4      | 4       | 26      |
| 4   | 160120805004 | AYESHA              | AB       | AB       | AB      | AB      | AB      | AB     | AB      | AB       | AB       | AB       | AB     | 0       | 0       |
| 5   | 160120805005 | B MOUNIKA           | 12       | 8        | 10      | 10      | 0       |        | 5       | 3        | 3        | 4        | 4      | 5       | 24      |
| 6   | 160120805006 | B SHYNISHA          | 17       | 1        | 9       | 10      | 10      | 8      | 10      | 1        | 3        | 4        | 4      | 5       | 28      |
| 7   | 160120805007 | CHAITRA GALI        | 16       | 16       | 16      | 0       | 8       | 8      | 8       | 2        | 3        | 5        | 4      | 5       | 33      |
| 8   | 160120805008 | C SUDHESHNA         | 17       | 18       | 18      | 10      | 10      | 9      | 10      | 5        | 5        | 3        | 5      | 5       | 38      |
| 9   | 160120805009 | DIVYA PREMA S       | 16       | 6        | 11      | 10      | 10      |        | 10      | 4        | 2        | 4        | 4      | 1       | 26      |
| 10  | 160120805010 | FOUZIA RAFATH       | 15       | 18       | 17      | 10      | 10      | 9      | 10      | 4        | 5        | 3        | 5      | 5       | 37      |
| 11  | 160120805011 | GAYATRI RAO         | 16       | 4        | 10      | 10      | 9       |        | 10      | 4        | 5        | 5        | 5      | 5       | 30      |
| 12  | 160120805012 | HAMSINI KATLA       | 17       | 6        | 12      | 10      | 9       | 8      | 10      | 2        | 4        | 4        | 4      | 5       | 31      |
| 13  | 160120805013 | JYOTHIKA K          | 17       | 16       | 17      | 10      | 9       |        | 10      | 4        | 5        | 5        | 5      | 5       | 37      |
| 14  | 160120805014 | KAVYA PASIRIKA P    | 16       | 2        | 9       | 0       | 8       | 8      | 8       | 5        | 5        | 3        | 5      | 5       | 27      |
| 15  | 160120805016 | SUJATHA K N V       | 16       | 8        | 12      | 10      | 8       |        | 9       | 5        | 3        | 3        | 4      | 5       | 30      |
| 16  | 160120805017 | NEHA REDDY M        | 17       | 6        | 12      | 10      | 9       | 7      | 10      | 4        | 4        | 4        | 4      | 5       | 31      |
| 17  | 160120805018 | PRAVALLIKA B        | 14       | 7        | 11      | 10      | 9       |        | 10      | 4        | 4        | 3        | 4      | 5       | 30      |
| 18  | 160120805019 | SAI LEELA SIRISHA V | 13       | 16       | 15      | 10      | 10      |        | 10      | 2        | 4        | 3        | 4      | 5       | 34      |
| 19  | 160120805020 | SAI SHRIYA Y        | 16       | 10       | 13      | 10      | 9       | 7      | 10      | 3        | 2        | 4        | 4      | 5       | 32      |
| 20  | 160120805021 | SANJANA REDDY P     | 17       | 7        | 12      | 10      | 10      | 8      | 10      | 2        | 5        | 4        | 5      | 5       | 32      |
| 21  | 160120805022 | SATHVIKA K          | 13       | 7        | 10      | 10      | 10      |        | 10      | 4        | 5        | 4        | 5      | 5       | 30      |
| 22  | 160120805023 | SHARVANI P          | 15       | 11       | 13      | 10      | 10      | 7      | 10      | 4        | 5        | 3        | 5      | 5       | 33      |
| 23  | 160120805024 | SHIVANI REDDY K     | 16       | 6        | 11      | 10      | 10      |        | 10      | 2        | 5        | 4        | 5      | 5       | 31      |
| 24  | 160120805025 | SHREECHANDRA S      | 15       | 4        | 10      | 10      | 10      | 8      | 10      | 4        | 2        | 2        | 3      | 5       | 28      |
| 25  | 160120805026 | SHREENIJA PERI      | 14       | 7        | 11      | 10      | 10      | 7      | 10      | 4        | 5        | 3        | 5      | 3       | 29      |
| 26  | 160120805027 | SHREYA BANALLA      | 13       | 1        | 7       | 10      | 9       | 8      | 10      | 4        | 4        | 3        | 4      | 5       | 26      |
| 27  | 160120805028 | SHRIYA REDDY P      | 16       | 8        | 12      | 10      | 9       | 7      | 10      | 3        | 5        | 3        | 4      | 5       | 31      |
| 28  | 160120805029 | SNEHA B             | 13       | 0        | 7       | 10      | 10      | 7      | 10      | 1        | 5        | 4        | 5      | 5       | 27      |

|    |              |                 |    |    |    |    |    |    |    |    |    |    |   |   |    |
|----|--------------|-----------------|----|----|----|----|----|----|----|----|----|----|---|---|----|
| 29 | 160120805030 | SOUBORNI NANDY  | 16 | 3  | 10 | 0  | 0  | 0  | 0  | 4  | AB | 4  | 4 | 0 | 14 |
| 30 | 160120805031 | SOUMYA MANDALA  | 16 | 6  | 11 | 10 | 8  |    | 9  | 2  | 3  | 4  | 4 | 5 | 29 |
| 31 | 160120805032 | SPOORTHI SADA   | 9  | 13 | 11 | 10 | 10 | 8  | 10 | 2  | 5  | 2  | 4 | 5 | 30 |
| 32 | 160120805033 | SRAVANI NEELAM  | 16 | 10 | 13 | 10 | 8  |    | 9  | 4  | 5  | 4  | 5 | 5 | 32 |
| 33 | 160120805034 | SRI VARSHA V    | 15 | 12 | 14 | 10 | 7  |    | 9  | 4  | 4  | 4  | 4 | 5 | 32 |
| 34 | 160120805035 | TANMAYI BOREDA  | 16 | 6  | 11 | 10 | 8  | 8  | 9  | 4  | 3  | 3  | 4 | 5 | 29 |
| 35 | 160120805036 | UMAMAH FATIMA S | 16 | 19 | 18 | 10 | 9  | 9  | 10 | 5  | 5  | 3  | 5 | 5 | 38 |
| 36 | 160120805037 | V SHREYA SHARMA | 18 | 17 | 18 | 10 | 10 | 9  | 10 | 5  | 5  | 5  | 5 | 5 | 38 |
| 37 | 160120805038 | VENNELA L       | 10 | 2  | 6  | 10 | 10 |    | 10 | 2  | 4  | 4  | 4 | 4 | 24 |
| 38 | 160120805039 | AKASH GADDAM    | 14 | 16 | 15 | 10 | 10 |    | 10 | 3  | 5  | 4  | 5 | 5 | 35 |
| 39 | 160120805040 | ALLOJU ABHISHEK | 11 | 7  | 9  | 10 | 9  | 8  | 10 | 3  | 3  | 4  | 4 | 4 | 27 |
| 40 | 160120805041 | SREERAM B       | 15 | 9  | 12 | 10 | 8  |    | 9  | 4  | 5  | 4  | 5 | 5 | 31 |
| 41 | 160120805042 | ASHISH R        | 17 | 3  | 10 | 10 | 8  |    | 9  | 2  | 4  | 4  | 4 | 5 | 28 |
| 42 | 160120805043 | B NITIN RATNAM  | 17 | 10 | 14 | 10 | 10 | 9  | 10 | 5  | 5  | 4  | 5 | 5 | 34 |
| 43 | 160120805044 | BALAJI DOOLAM   | 15 | 16 | 16 | 10 | 8  | 10 | 10 | 5  | 4  | 4  | 5 | 5 | 36 |
| 44 | 160120805045 | BHANU PRAKASH T | 13 | 5  | 9  | 10 | 10 | 8  | 10 | 3  | 4  | 3  | 4 | 5 | 28 |
| 45 | 160120805046 | CHENNA CHARAN M | 14 | 14 | 14 | 10 | 8  |    | 9  | 3  | 4  | 3  | 4 | 5 | 32 |
| 46 | 160120805047 | DINESH REDDY P  | 13 | 2  | 8  | 10 | 8  |    | 9  | 1  | 5  | 4  | 5 | 2 | 24 |
| 47 | 160120805048 | DIVYAMSHU S     | 8  | 5  | 7  | 10 | 7  |    | 9  | 5  | 4  | AB | 5 | 4 | 25 |
| 48 | 160120805049 | GOURAV T        | 13 | 2  | 8  | 10 | 9  | 8  | 10 | 2  | 3  | 3  | 3 | 4 | 25 |
| 49 | 160120805050 | HARISH POLE     | 11 | 3  | 7  | 10 | 10 |    | 10 | 1  | 3  | 3  | 3 | 4 | 24 |
| 50 | 160120805051 | HRITHIK KOLLURU | 17 | 7  | 12 | 10 | 8  | 8  | 9  | 2  | 4  | 5  | 5 | 5 | 31 |
| 51 | 160120805052 | KCHETAN BABU    | 16 | 9  | 13 | 10 | 10 | 8  | 10 | 4  | 5  | 3  | 5 | 5 | 33 |
| 52 | 160120805053 | M VIKKI KUMAR   | 9  | 4  | 7  | 8  | 7  | 0  | 8  | AB | 5  | 4  | 5 | 0 | 20 |
| 53 | 160120805054 | M CHANDRA MADAS | 16 | 2  | 9  | 10 | 10 | 9  | 10 | 4  | 4  | 4  | 4 | 5 | 28 |
| 54 | 160120805055 | RAKESH REDDY N  | 10 | 6  | 8  | 10 | 6  |    | 8  | 2  | 3  | 4  | 4 | 4 | 24 |
| 55 | 160120805056 | SAI CHANDRA K   | 14 | 4  | 9  | 6  | 0  | 8  | 7  | AB | 4  | 3  | 4 | 0 | 20 |
| 56 | 160120805057 | SAI PRATHIB K   | 15 | 7  | 11 | 10 | 10 |    | 10 | 4  | 4  | 4  | 4 | 5 | 30 |
| 57 | 160120805058 | SAMANTH C       | 15 | 17 | 16 | 10 | 8  | 7  | 9  | 4  | 5  | 4  | 5 | 5 | 35 |
| 58 | 160120805059 | SUMANTH RAO M   | 9  | 0  | 5  | 0  | 8  | 7  | 8  | 3  | 4  | 3  | 4 | 5 | 22 |
| 59 | 160120805060 | YASHASVI K      | 16 | 1  | 9  | 10 | 10 | 0  | 10 | 2  | 5  | 2  | 4 | 5 | 28 |


  
 Faculty Signature



**Course Code: 20CS C01**

**Course Title: PROGRAMMING FOR PROBLEM SOLVING**

**CO-PO Articulation matrix- Target**

3.1.3 Course Articulation Matrix

|     | PO1   | PO2   | PO3   | PO4   | PO5   | PO6   | PO7 | PO8 | PO9 | PO10  | PO11  | PO12  | PSO1  | PSO2  |
|-----|-------|-------|-------|-------|-------|-------|-----|-----|-----|-------|-------|-------|-------|-------|
| CO1 | 2     | 1     | 2     | -     | -     | -     | -   | -   | -   | 1     | 1     | 2     | 2     | 2     |
| CO2 | 3     | 3     | 3     | 2     | -     | 1     | -   | -   | -   | 1     | 1     | 2     | 2     | 3     |
| CO3 | 2     | 2     | 2     | 1     | -     | -     | -   | -   | -   | 1     | -     | 2     | 2     | 2     |
| CO4 | 3     | 3     | 3     | 2     | 3     | -     | -   | -   | -   | 1     | 1     | 2     | 2     | 3     |
| CO5 | 3     | 2     | 2     | 1     | 2     | -     | -   | -   | -   | 1     | -     | 2     | 2     | 3     |
| CO6 | 3     | 3     | -     | -     | 3     | -     | -   | -   | -   | 3     | -     | 3     | 3     | 3     |
| Avg | 2.667 | 2.333 | 2.400 | 1.500 | 2.250 | 1.000 | -   | -   | -   | 1.333 | 1.000 | 2.167 | 2.333 | 2.667 |

| PD Attainment | Level | Target Level for Course |      |      |     |      |      |     |     |     |     |      |      |      |
|---------------|-------|-------------------------|------|------|-----|------|------|-----|-----|-----|-----|------|------|------|
|               |       | 50                      |      |      |     |      |      |     |     |     |     |      |      |      |
| x >= 70       | 3     | (2020-21) PPS           | PO1  | PO2  | PO3 | PO4  | PO5  | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
| 60 <= x < 70  | 2     |                         | 1.13 | 1.17 | 1.2 | 0.75 | 1.13 | 0.5 | -   | -   | -   | 0.67 | 0.5  | 1.08 |
| 50 <= x < 60  | 1     | Direct Attainment       | -    | -    | -   | -    | -    | -   | -   | -   | -   | -    | -    | -    |
| x <= 40       | 0     | Gap                     | -    | -    | -   | -    | -    | -   | -   | -   | -   | -    | -    | -    |

INSTRUCTOR

HEAD DEPT OF CSE  
Professor and Head Department  
Department of Computer Science & Engineering  
Chaitanya Bharathi Institute of Technology (A)  
Gandipet, Hyderabad-500 075.(T.S.)