

---

# Programming for Problem Solving

1st Year 1st Semester 2022-23

Credits: 3

## 1 Mark Question and Answers

1. Define Algorithm
2. Characteristics of Algorithms.
3. Adv, dis adv algorithm.
4. Define flow chart
5. Symbols flowchart
6. What is variable
7. How to declare, a variable,
8. Identifier rules
9. What is Compiled
10. What is type conversion
11. What is storage class
12. What is an Array
13. Syntax for multidimensional
14. diff between while & do while
15. diff between ++a & a++
16. syntax for Conditional operator
17. Discuss Strlen() with Example
18. what is String A
19. Syntax for Array of string
20. what is the use for '\0'
21. Define Structure
22. Define union.
23. Mention the difference between Structure & union?
24. What is Pointer, Adv & dis Adv Pointer
25. How the pointer variable Declared
26. what do you mean by array of pointers
27. Explain about the structures containing Pointer with Example
28. Explain about self referential struct
29. Syntax for enum

- 
30. What is Preprocess?
  31. Diff blw undef & ifndef
  32. what is file
  33. Syntan for fputc(), fgetc( )
  34. Syntax for fread() fwrite()
  35. Syntax for fprintf(), fscanf()
  36. Syntax for fputs(), fgets()
  37. write the use of the rewind function.
  38. what is binary file
  39. Explain. modes
  40. What is function
  41. Adv of function
  42. Uses of function.
  43. Define formal Arguments
  44. Limitation of recursion.
  45. Advantage and Disadvantages recursion
  46. Syntax for malloc, calloc, realloc.
  47. Syntax for Pointer to function.
  48. Explain about Parameter Passing Using copy by address.
  49. what is time complexity
  50. Define best case Performance of an algorithm.
  51. Time Complexity for linear, bubble search; bubble, selection, Insertion sort.
    - a. Best case
    - b. Average case
    - c. Worst case

---

## 1. Define Algorithm

- a. Algorithm is a finite sequence of instructions, each of which has a clear meaning and can be performed with a finite amount of effort in a finite length of time. No matter what the input values may be, an algorithm terminates after executing a finite number of instructions.
- b. The ordered set of instructions required to solve a problem is known as an algorithm.

## 2. Characteristics of Algorithms.

- a. Precision – the steps are precisely stated (defined).
- b. Uniqueness – results of each step are uniquely defined and only depend on the input and the result of the preceding steps.
- c. Finiteness – the algorithm stops after a finite number of instructions are executed.
- d. Input – the algorithm receives input.
- e. Output – the algorithm produces output.
- f. Generality – the algorithm applies to a set of inputs.

## 3. Advantages, Disadvantages algorithm.

### a. Advantages of Algorithms:

- i. It is a stepwise representation of a solution to a given problem, which makes it easy to understand.
- ii. An algorithm uses a definite procedure.
- iii. It is not dependent on any programming language, so it is easy to understand for anyone even without programming knowledge.
- iv. Every step in an algorithm has its own logical sequence so it is easy to debug.
- v. By using algorithms, the problem is broken down into smaller pieces or steps hence; it is easier for programmers to convert it into an actual program.

### b. Disadvantages of Algorithms:

- i. Algorithms is Time consuming.
- ii. Difficult to show Branching and Looping in Algorithms.
- iii. Big tasks are difficult to put in Algorithms.

c.









## 4. Define flow chart

- a. Flowchart is a diagrammatic representation of an algorithm. Flowchart is very helpful in writing programs and explaining programs to others.

## 5. Symbols flowchart

- a. Different symbols are used for different states in flowchart, for example: Input/Output and decision making has different symbols. The table below describes all the symbols that are used in making

## flowchart

Symbol	Purpose	Description
	Flow line	Used to indicate the flow of logic by connecting symbols.
	Terminal(Stop/Start)	Used to represent start and end of flowchart.
	Input/Output	Used for input and output operation.
	Processing	Used for arithmetic operations and data-Manipulations.
	Decision	Used to represent the operation in which there are two alternatives, true and false.
	On-page Connector	Used to join different flow line
	Off-page Connector	Used to connect flowchart portion on different page.
	Predefined Process/Function	Used to represent a group of statements performing one processing task.

### 6. What is variable

- In C language, a variable is a named storage location in the computer's memory that is used to hold a value. It is an essential part of any program and provides a way to store and manipulate data during the execution of the program.

### 7. How to declare, a variable,

- A variable in C must be declared before it is used in the program. The declaration specifies the data type of the variable (such as int, float, char, etc.) and the variable name. For example, to declare an integer variable named "num", the following statement can be used:

**`int num;`**

### 8. Identifier rules

- There are certain rules that should be followed while naming c identifiers:
  - They must begin with a letter or underscore (\_).
  - They must consist of only letters, digits, or underscore. No other special character is allowed.
  - It should not be a keyword.
  - It must not contain white space.
  - It should be up to 31 characters long as only the first 31 characters are significant.

### 9. What is Compiler

- The C compiler translates source to assembly code. The source code is received from the preprocessor.

### 10. What is type conversion

- 
- a. When variables and constants of different types are combined in an expression then they are converted to same data type. The process of converting one predefined type into another is called type conversion.

#### 11. What is storage class

- a. Storage class refers to the scope and lifetime of a variable, which determines how the memory for the variable is allocated, accessed, and released during program execution.

#### 12. What is an Array

- a. An array is a collection of similar data elements of the same data type, stored in contiguous memory locations. Each element in an array is identified by an index, which represents its position in the array.

#### 13. Syntax for multidimensional Array

- a. *data\_type array\_name[size1][size2]...[sizeN];*

#### 14. diff between while & do while

- a. The main difference between while and do-while loops in C language is in the way they check for the loop condition:
  - i. while loop: The while loop first checks the loop condition before executing the loop body. If the condition is true, then the loop body is executed, and the condition is checked again. The loop continues until the condition becomes false.
  - ii. do-while loop: The do-while loop first executes the loop body, and then checks the loop condition. If the condition is true, then the loop body is executed again, and the condition is checked again. The loop continues until the condition becomes false.

#### 15. diff between ++a & a++

- a. ++a (pre-increment): This operator increments the value of a by 1 and then returns the new value of a.
- b. a++ (post-increment): This operator returns the current value of a and then increments a by 1.

#### 16. syntax for Conditional operator

- a. *condition ? expression1 : expression2;*
- b. Here, condition is a boolean expression that evaluates to either true or false. If condition is true, then expression1 is evaluated and returned as the result of the entire expression. If condition is false, then expression2 is evaluated and returned as the result of the entire expression.

#### 17. Discuss Strlen() with Example

- a. This function is used to find the length of the string excluding the NULL character. In other words, this function is used to count the number of characters in a string. Its syntax is as follows:

*int strlen(string);*

**Example:**     `char str1[ ] = "WELCOME";`  
                  `int n;`  
                  `n = strlen(str1);`

## 18. what is String

- In C language a string is group of characters (or) array of characters, which is terminated by delimiter \0 (null).
- `char str[6] = {'h', 'e', 'l', 'l', 'o', '\0'};`  
`char str[] = "hello";`

## 19. Syntax for Array of string

- `char string_name[size];`
- `char str[6] = {'h', 'e', 'l', 'l', 'o', '\0'};`
- `char str[] = "hello";`

## 20. what is the use for '\0'

- The last character in the array must be the null character '\0', which indicates the end of the string.

## 21. Define Structure

- A structure is a collection of one or more variables of different data types, grouped together under a single name. By using structures variables, arrays, pointers etc can be grouped together.

## 22. Define union.

- A union is one of the derived data types. Union is a collection of variables referred under a single name. The syntax, declaration and use of union is similar to the structure but its functionality is different.

## 23. Mention the difference between Structure & union?

- 

	Structure	Union
Keyword	Struct	Unio
Definition	A structure is a collection of logically related elements, possibly of different types, having a single name.	A union is a collection of logically related elements, possibly of different types, having a single name, shares single memory location.
Declaration	<pre>struct tag_name{     type1 member1; type1     member2; ..... }; struct tag_name var;</pre>	<pre>union tag_name{     type1 member1; type1     member2; ..... }; union tag_name var;</pre>
Initialization	Same	Same
Accessing	Accessed by specifying structure_ variable_name.member_name	Accessed by specifying union_ variable_name.member_name
)Memory Allocation	Each member of the structure occupies unique location, stored in contiguous locations.	Memory is allocated by considering the size of the largest member. All the members share the common

		location.
Size	Size of the structure depends on the type of members, adding size of all the members. <code>sizeof (st_var);</code>	Size is given by the size of the largest member <code>sizeof(un_variable)</code>
Using pointers	Structure members can be accessed by using dereferencing operator dot and selection operator(->)	same as structure
	We can have arrays as a member of structures. All members can be accessed at a time.	We can have array as a member of union. Only one member can be accessed at a time.
	Nesting of structures is possible.	same.
	It is possible structure may contain union as a member.	It is possible union may contain structure as a member

## 24. What is Pointer, Adv & dis Adv Pointer

- a. Pointer is a user defined data type that creates special types of variables which can hold the address of primitive data type like char, int, float, double or user defined data type like function, pointer etc. or derived data type like array, structure, union, enum.
- b. Advantages
  - i. Pointers are more efficient in handling arrays and data tables.
  - ii. Pointers can be used to return multiple values from a function via function arguments.
  - iii. The use of pointer arrays to character strings results in saving of data storage space in memory.
  - iv. Pointers allow C to support dynamic memory management.
  - v. Pointers provide an efficient tool for manipulating dynamic data structures such as structures, linked lists , queues , stacks and trees.
  - vi. Pointers reduce length and complexity of programs.
  - vii. They increase the execution speed and thus reduce the program execution time.
- c. Disadvantages
  - i. Pointers can be difficult to understand and use correctly, leading to programming errors and bugs.
  - ii. Pointer arithmetic can be error-prone, causing problems like buffer overflows and memory leaks.
  - iii. Pointers can make programs less secure, since they can be used to manipulate memory outside of a program's allocated space.

- iv. Pointer manipulation can make programs harder to read and maintain, especially when dealing with complex data structures.
- v. Pointers are not available in all programming languages, which can make it harder to port code to other platforms or languages.

## 25. How the pointer variable Declared

- a. `data_type *pointer_variable_name;`
- b. `int *ptr;`

## 26. what do you mean by array of pointers

- a. An array of pointers is an array in which each element is a pointer to a memory location where another value (of the same or different data type) is stored.

## 27. Explain about the structures containing Pointer with Example

- a. Pointers pointing to structures are called structure pointers.
- b. This allows us to create self-referential structures, which are structures that have a member that points to another instance of the same structure type.
- c. Example:

```
struct Node {  
    int data;  
    struct Node *next;  
};
```

## 28. Explain about self referential struct

- a. A structure can contain a pointer as one of its members. This allows us to create self-referential structures, which are structures that have a member that points to another instance of the same structure type.

## 29. Syntax for enum

- a. `enum enum_name { enumerator1, enumerator2, ..., enumeratorN };`
- b. `enum weekdays { Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday };`

## 30. What is Preprocess?

- a. Before a C program is compiled in a compiler, source code is processed by a program called preprocessor. This process is called preprocessing.
- b. Commands used in preprocessor are called preprocessor directives and they begin with “#” symbol.

## 31. Diff between undef & ifndef

- a. `undef`: This directive is used to remove a previously defined macro.
- b. `ifndef`: This directive is used to check whether a macro is already defined or not.

## 32. what is file

- a. a file is a collection of related data that is stored in secondary memory (such as a hard disk) and can be accessed by a program when needed.
- b. There are two types of files in C: Text files, Binary files. files can be accessed using file pointers.



---

### 33. Syntan for fputc(), fgetc()

- a. `int fputc(c, file_pointer);`
- b. `int fgetc(file_pointer);`

### 34. Syntax for fread() fwrite()

- a. `fread(&structure_variable ,sizeof(structure_variable), 1, file_pointer);`
- b. `fwrite(&structure_variable ,sizeof(structure_variable), 1, file_pointer);`

### 35. Syntax for fprintf(), fscanf()

- a. `fprintf(fp,"control string",list);`
- b. `fscanf(fp,"control string",list);`

### 36. Syntax for fputs(), fgets()

- a. `fputs(string, file_pointer);`
- b. `fgets(string, length, file_pointer);`

### 37. write the use of the rewind function.

- a. It takes a file pointer and resets the position to the start of the file.
- b. `rewind(fp);`

### 38. what is binary file

- a. These are files that contain data in a format that is not human-readable, such as executable files, images, videos, or audio files.

### 39. Explain modes

- a. The following are the different modes in which you can open a file in C:
  - i. **"r" (Read mode):** This mode is used to open an existing file for reading purposes. In this mode, you cannot modify the contents of the file. If the file does not exist, an error message is displayed.
  - ii. **"w" (Write mode):** This mode is used to create a new file or to overwrite an existing file. In this mode, you can modify the contents of the file. If the file does not exist, a new file is created.
  - iii. **"a" (Append mode):** This mode is used to open an existing file and add new content to it. In this mode, you cannot modify the existing content of the file. If the file does not exist, a new file is created.
  - iv. **"r+" (Read and Write mode):** This mode is used to open an existing file for both reading and writing purposes. In this mode, you can modify the contents of the file.
  - v. **"w+" (Write and Read mode):** This mode is used to create a new file or to overwrite an existing file. In this mode, you can read and modify the contents of the file.
  - vi. **"a+" (Append and Read mode):** This mode is used to open an existing file and add new content to it. In this mode, you can also read the existing content of the file.

---

#### 40. What is function

- a. A function is a block of code that performs a specific task. It has a name and it is reusable .It can be executed from as many different parts in a program as required; it can also return a value to calling program.

#### 41. Adv of function

- a. Reusability: Functions allow for code reuse, as a function can be called multiple times from different parts of the program, without the need to rewrite the code.
- b. Modularization: Functions enable the program to be broken down into smaller, more manageable modules, making it easier to read, understand, and maintain.
- c. Code Optimization: Functions allow for code optimization, as repetitive tasks can be encapsulated into a function, reducing the amount of code and improving the overall efficiency of the program.
- d. Easy Debugging: Functions help in debugging the program by allowing us to isolate problems in a specific part of the code.
- e. Abstraction: Functions can hide the complexity of the code, by providing a simpler interface to the user, making it easier to use and understand.

#### 42. Uses of function.

- a. Parameter passing: Functions can take arguments as inputs and return values as outputs, allowing for flexibility in how data is processed and manipulated.
- b. Code organization: Functions help in organizing code into logical sections, making it easier to debug and modify.
- c. Library functions: C comes with a large number of predefined functions that are part of the standard library, providing useful functionality that can be easily used in programs.

#### 43. Define formal Arguments

- a. Formal arguments refer to the input parameters that are declared in the function prototype. They are also known as formal parameters. These arguments are specified in the function declaration and are used to pass values to the function when it is called.

#### 44. Limitation of recursion.

- a. Recursive solution is always logical and it is very difficult to trace.(debug and understand).
- b. In recursive we must have an if statement somewhere to force the function to return without the recursive call being executed, otherwise the function will never return.
- c. Recursion takes a lot of stack space, usually not considerable when the program is small and running on a PC.
- d. Recursion uses more processor time.

#### 45. Advantage and Disadvantages recursion

- a. Advantages:
  - i. Recursive functions can simplify the code and make it easier to understand.
  - ii. They can break a complex problem into smaller subproblems and solve them individually.

- iii. Recursive functions can save memory by reusing the same code instead of creating new functions.
- b. Disadvantages:
  - i. Recursive functions can be less efficient than their iterative counterparts due to the overhead of function calls.
  - ii. Recursive functions can cause stack overflow if they are called too many times or if the base case is not properly defined.
  - iii. Debugging recursive functions can be more difficult than debugging iterative functions.
  - iv. Recursive functions can lead to infinite recursion if the base case is not properly defined, causing the program to crash or enter an infinite loop.

**46. Syntax for malloc, calloc, realloc.**

- a. `ptr = (cast-type*) malloc(byte-size);`
- b. `ptr = (cast-type*) calloc(num, size);`
- c. `ptr = realloc(ptr, new-byte-size);`

**47. Syntax for Pointer to function.**

- a. 

```
void function_name(data_type* variable_name) {  
    // Function body  
}
```

**48. Explain about Parameter Passing Using copy by address.**

- a. Parameter Passing Using copy by address is a method of passing parameters to a function in C language. In this method, the address of the actual argument is passed to the function.

**49. what is time complexity**

- a. The amount of time taken by an algorithm to execute its instructions. It is often denoted using Big O notation, which describes the upper bound of the time taken by an algorithm to complete its task as a function of the input size.

**50. Define best case Performance of an algorithm.**

- a. The "best case performance" of an algorithm refers to the minimum amount of resources (time or space) that an algorithm requires for a given input size when everything goes well. It represents the fastest possible time or the smallest possible space needed to complete the task.

**51. Time Complexity for linear, bubble search; bubble, selection, Insertion sort.**

- a. **Best case**
- b. **Average case**
- c. **Worst case**
  - i. Linear search:
    - 1. **Best case:  $O(1)$**  (when the element to be searched is present at the first index)
    - 2. **Average case:  $O(n/2)$**
    - 3. **Worst case:  $O(n)$**  (when the element to be searched is not present in the array)
  - ii. Binary Search:

- 
1. **Best case:  $O(1)$**  (when the element to be searched is present at the middle index)
  2. **Average case:  $O(\log n)$**
  3. **Worst case:  $O(\log n)$**

iii. Bubble Sort:

1. **Best case:  $O(n)$**  (when the array is already sorted)
2. **Average case:  $O(n^2)$**
3. **Worst case:  $O(n^2)$**

iv. Selection Sort:

1. **Best case:  $O(n^2)$**
2. **Average case:  $O(n^2)$**
3. **Worst case:  $O(n^2)$**

v. Insertion Sort:

1. **Best case:  $O(n)$**  (when the array is already sorted)
2. **Average case:  $O(n^2)$**
3. **Worst case:  $O(n^2)$**