Assignment No:-3

Q1] What are the different standardizations of C programming?

→ The C programming language has various standardizations to ensure consistency and portability across different systems and compiler.

① K&R (1978) :-

This was the earlist and most widely used version before formal standardization.

② ANSI C (C89/C90) :-

C89 - Standardized by the American National Standards Institude in 1989.

C90 - The same standard was adopted by the International (ISO) in 1990.

- This standard introduced several features, including function prototypes, the void type and more stringent type checking

③ C-99 (ISO/IEC 9899-1999) :-

This standard introduced several improvements such as:

- variable-length arrays.
- Inline functions.
- New data types like long int, bool and complex numbers.
- improved support for floating-point operations.
- Single-line Comments using //, among other featur

④ C23 (ISO/IEC 9899:2023)

- The latest standard published in 2023. It introduces enhanced safety features, such as safer memory functions.
- Expanded floating-point and math support.

Q2] What is mean by Data Structures.

→ A data structure refers to a specific way of organizing, managing and storing data in the computer's memory so that it can be accessed and manipulated efficiently.

- Data structures are fundamental for creating efficient programs, and they form the backbone of algorithms used in software development.
- The concept of data structure is language Independent.
- The ~~to~~ concept of data structure is ~~implem~~ applicable in every programming language. (C, C++, Java)
- Data Structures can be broadly classified into two main types:

Linear - data structure

Non- Linear data Structure

① Linear Data Structure-

In linear data structure elements are arranged in sequencial manner

- Each element in a sequencial manner and each element is connected to its previous and next element.
- The data stored in linear order, and operations typically follow this order.
  - Array
  - Linked list
  - Stacks
  - Queues

② Non-linear Data Structure :

- In non linear data structures, elements are not arranged in a sequencial order.
- Instead, they are arranged in a hierarchical manner, with each element possibly connected to multiple other elements.
  - Trees
  - Graphs
  - & Hash Tables

Q3] What is Array, explain in detail.

→ - An array is collection of elements, all of the same type, stored in contiguous memory locations.
- Arrays allow to store multiple values of the same data type using a single variable name.
- Making it easier to manage and manipulate large sets of data.
- Array considered as a linear data structure.
Array is cosidered as derived data type in C, C++, Java.
- It is derived data type which holds multiple homogenuous elements in index format.

• Declaring and Initializing Arrays

• Declaration :-

To declare an Array, specify the data type of its elements, the array name, and the size

e.g. int arr [0];

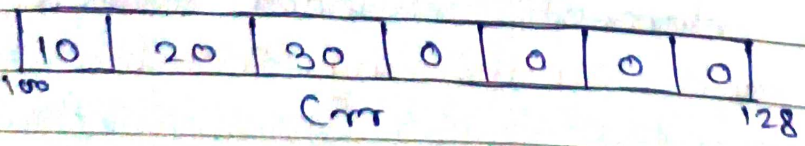
• Initialization

There are multiple ways in which we can create and initialized the array.

① With Size and Initialization List:
Create the array by specifing the length of array as well as initializing the array imediately using initialization of array.
Syntax-
datatype arrayname [length] = {value1, ... valuen};
eg -
int Arr [5] = {10, 20, 30, 40, 50};

② Array declaration without length.
we can also let the compiler determine the size of the array based on number of elements provided.
Syntax-
datatype arrayname [] = {value1, ... valuen};
e.g.
int Brr [] = {10, 20, 30, 40}

③ Initializing of array with less numbers.
If we don't specify all elements, the remaining elements are initialized to zero (default value of int)
e.g.
int Crr [7] = {10, 20, 30};



④ Initialization after declaration.
e.g. int Drr [3];
Drr [0] = 10;
Drr [1] = 20;
Drr [2] = 30;

• Properties of Array.

• Homogeneous Elements - All elements in an array must be of the same data type.

• Contiguous Memory Allocation - Elements of the array are stored in consecutive memory locations.

• Fixed Size - The size of an array is specified at the time of its declaration and cannot be changed during runtime.

• Zero-based Indexing - Array indexing starts at 0, meaning the first element is accessed with index 0, the second element with index 1, and so on.

Q4] Differentiate the Constant and Variable.

→

Variables	Constant
• A variable is a storage location in memory with a specific name and type, where data can be stored, modified and retrieved during the execution of program.	• Constant is a storage location in memory with a specific name and type, where data is stored but cannot be modified after it has been initialized.
• Variables are mutable, meaning their values can be changed at any point in program after they have been initialized.	• Constants are immutable, meaning their values cannot be changed once they have been initialized.

Variables	Constants

• Variables are declared with a data type, and they can be initialized with a value.
e.g. int x = 10;
x = 20;
	• Constants are declared with the const keyword, and they must be initialized at the time of declaration.
	e.g. const int y = 10;
	y = 20; // not Allow

• Variables are used when the value needs to change during the execution of program.	• Constants are used when you need to ensure that a value remains un-changed throughout the program.

Q5] Difference between Local & Global Variables.

Local	Global
• Local variables are variables that are declared inside a function, block or a compound statement.	• Global variables are variables that are declared outside of all functions, typically at the top of the program file.
• The scope of a local variable is limited to the function or block where it is declared. It cannot be accessed outside that function or block.	• The scope of global variable is entire program, meaning it can be accessed by any function within the program file.

Local	Global
• The lifetime of local variable is limited to the duration of the function or block in which it is defined. The variable is created when the function/ block is entered and destroyed when the function/block is exited.	• The lifetime of global variable is the duration of the program's execution. It is created when the program starts and destroyed when the programs ends.
• Local variables are typically stored on the stack	• Global are stored in a fixed location in memory
• If local variable is not explicitly initialized, it contain a garbage value.	• Global are automatically initialized to zero if not explicitly initialized.

Q6] Explain the concept of function, explain about the starting point function in C, C++, Java.

→ A function is a block of code that performs a specific task. Functions help organize code by dividing by program into smaller, reusable pieces. This improves readability, maintainability and allow code reuse across the program or other program

• Function Declaration

A function must be declared before it is used. The declaration tells the compiler the function's name, its return type, and the types of its parameter

• Syntax

returntype function_name(parametertype1, ... parameter-typen);

eg. int add (int a, int b);

• Function Definition:

This is the actual code that defines the function and specifies what it does. It includes the function body where the logic is written.

• Syntax:

```
return_type fun_name (parameter1, .... parametern){
  // function body
  return value;
}
```

• eg

```
int add (int a; int b) {
   return a+b;
```

• Function Call:

To execute a function, you call it by its name and provide the necessary argument (if any). This makes the function perform its task.

• Syntax

```
function_name (argument1,..argumentn);
```

• eg.

```
int result = add (5, 10);
```

• Return Type:

The return type indicates what kind of value a function will return after its execution. If a function does not return a value, its return type is void.

```
eg. int add (int a, int b);
      void &display ();
```

• Parameters and Arguments:

Parameters are variables listed in the function declaration and definition that receive the

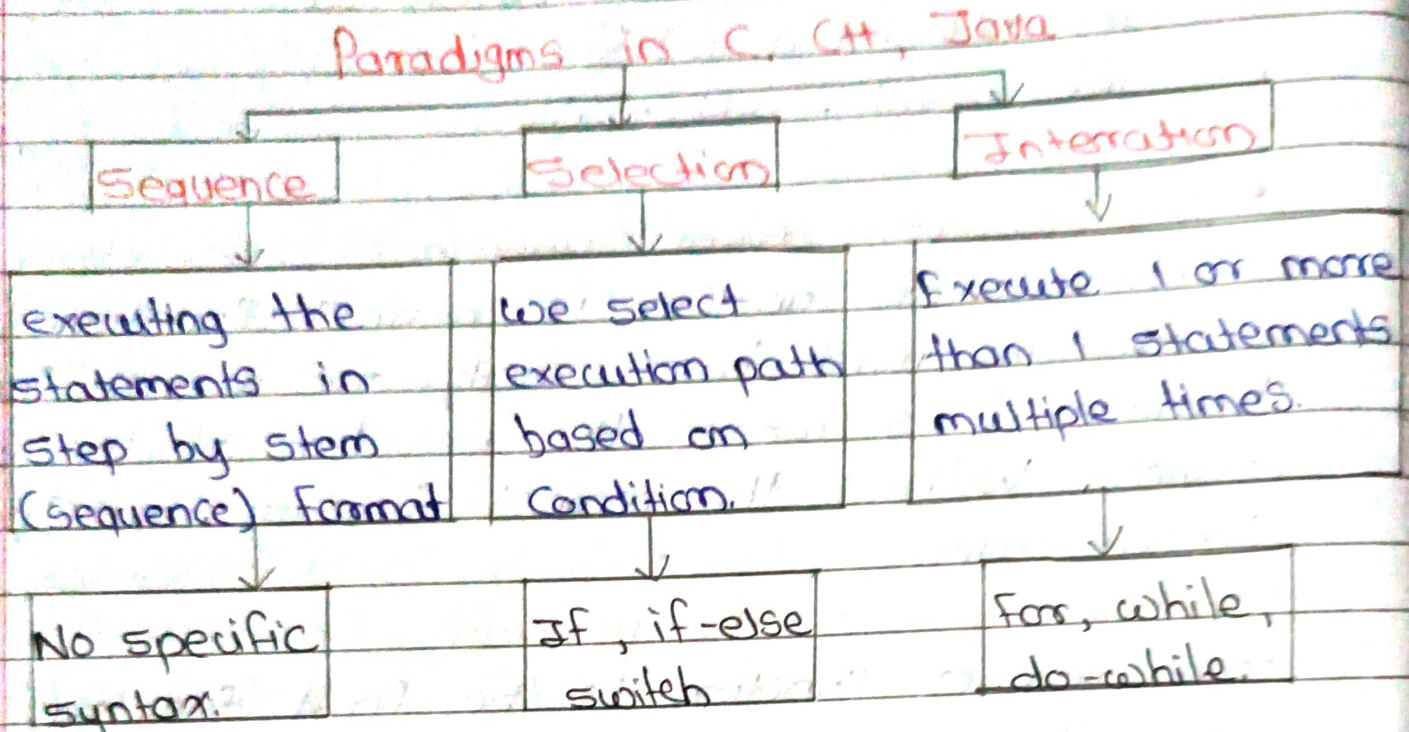
values passed into the function.
Arguments are the actual values you pass to the function during a function call.
e.g. int add (int a, int b);
int result = add (3,4);.

• The starting point function
starting point function is main() function. Every C program must have exactly one main() function because it is where the execution of the program begins. Without the main() function, a C program cannot run.

- Entry point of the program.
The main() function is the first function called when the program is executed. Regardless of how many other functions a program may have, the execution always starts with main() function.

- Return value of main().
The main() function returns an integer value, which is used by the operating system to determine whether the program executed successfully.
return 0 - Usually returned if the program ran successfully.
return 1 - Used to signal an error or abnormal termination.

Q 7 Explain about the Programming language Paradigms in C, C++, Java.



1] Sequence - It is simplest control structure where statements are executed one after another in order they appear in the program.
- There is no branching or repeating in this struct
It forms the basic building block of any program

e.g.

```
#include <stdio.h>
int main(){
   int a = 5;
   int b = 10;
   int sum = a+b;
   printf("sum: %d\n", sum);
   return 0;
}
```

2] Selection (Decision Making) - Allows the program

to make decisions on execute certain parts of code based on conditions.
- This is typically done using if, if-else, switch and similar structures.
e.g.

```
#include <stdio.h>
int main(){
   int (a>5)
   int a = 5;
   if (a>0){
      printf(" a is a positive number\n");
   }else{
      printf(" a is negative number\n");
   }
   return 0;
}
```

3) Iteration (Loops) - It is used when you need a repeat a block of code multiple times.
- This is accomplished using loops.
- the common loops used in C, C++, Java are for, while, and do while.
eg. (For loop)

```
#include <stdio.h>
int main(){
   for(int i =0; i<5; i++){
      paint("i = %d\n", i);
   }
   return 0;
}
```

Q8] Write a program to accept a input from user and find whether the no is even or odd, draw the diagrammatic layout of some program

→

```
#include <stdio.h>
int main () {
    int num;
    printf ("Enter a number : \n");  //input from user.
    scanf (" %d", &num);

    if (num % 2 == 0) {
        printf ("%d is even number \n", num);
    } else {
        printf ("%d is odd number \n", num);
    }
    return 0;
}
```

sequence

selection

Diagramatic Layout.

No 0	num%2==0 0
No 4	num %2==0 0
0==0 True number is even.	

No 0	num %2 == 0 1
No 5	num %2 == 0 1
0 == 1 False number is odd.	

If-else statement is used to choose between two paths based on the condition.
- If the condition is true, one block is executed, another block is executed fore else statement.

Q9] Write short note on short hand operator.

→ In C, C++, Java increment and decrement operators are used to increase or decrease the value of a variable by 1.
- These operators come in two forms.
Pre - increament, Post - increment.
Pre - decrement, Post - decrement.

• Pre - increment - (++i)
The variable is increment first, then it updated value is used in the expresion.
No = ++i; No = 10, i = 10

NO [10 11] First increment
i [10 11] Then assign

• Post Increment - (i++)
The variable's original value is used in the expression, then it is incremented.
No = i++;

NO [10 10] First assign
i [10 11] then increment.

• Pre - decrement - (--i)
The variable is decremented first, then its updated value is used in the expression.
No = --i;

NO [10 9] first decrement
i [10 9] then assign.

Post decrement - (i--)
The variable's original value is used in the expression and then it is decremented.
No ~~to~~ 10 first assignn.
i ~~to~~ 9 the decrement.

Q.10] Write down notes on various loops its syntax and ~~explain~~ examples along with the dry run diagram.

→ loops allows you to execute a block of code repeatedly, based on the conditions.
There are three main types of loops in most programming languages like C, C++ and Java.
For loop
while loop
do-while loop

1] For loop- Used when you know the number of iterations in advance.
It consist of four parts.
- Initialization - Initialize the loop control variable
- Condition - The loop runs while the condition is true.
- Incre/Decrement - The loop control variable is updated after every iteration.
- loop body - content which is putput of program.

Syntax-
```
for (initialization; Condition; Incre/decrement) {
    // code to be executed
}
```

e.g.→

```
#include <stdio.h>
int main()
{
    int i = 0;
    for (i=1 ; i<=5 ; i++)
    {
      printf("%d. Jay Ganesha.... \n", i);
    }
    return 0;
}
```

Dry Run -

Iteration I -

1: i = 1

2: i <= 5 — True

4: Jay Ganesha...

3: i++ [+ 2] (i)

Iteration II -

i = 2

2: 2 <= 5 — True

4: Jay Ganesha...

3: i++ [2 3] (i)

Iteration III -

i = 3

2: 3 <= 5 — True

4: Jay Ganesha...

3: i++ [3 4] (i)

Iteration IV

i = 4

2: 4 <= 5 — True

4: Jay Ganesha...

3: i++ [4 5] (i)

Iteration V

i = 5

2: 5 == 5 — True

4: Jay Ganesha...

3: i++ [5 6]

Iteration 6:

i = 6

2: 6 <= 5 — False

break

2] While loop - is used when you don't know the number of iterations in advance.
the loop continues as long as the condition is true.

Syntax

```
while (condition) {
    // code to be executed
}
```

e.g.

```
#include <stdio.h>
int main()
{
  int i=1;
  while (i<=5)
  {
    printf ("%d Jay Ganesha ... \n", i);
    i++;
  }
return 0;
}
```

Dry Run

Iteration I
1: i = 1 [x2]
2: 1<=5 - true
4: 1. Jay Ganesha...
3: i++

Iteration II [23]
2: 2<=5 - True
4: 2. Jay Ganesha...
3: i++

Iteration III [34]
2: 3<=5 - True
4: 3. Jay Ganesha...
3: i++

Iteration IV [43]
2: 4<=5 - True
4: 4. Jay Ganesha...
3. i++

Iteration 5 [6]
2: 5<=5 - True
4: 5. Jay Ganesha...
3: i++

Iteration 6
2: 6<=5 - False
Break.

3] Do-while loop - is similar to the while loop, but the condition is checked after executing the block of code.
- This ensures that the loop runs at least once.
- In case of do while loop the condition gets checked at the end of the loop.
- Irrespective of checking the condition the do-while loop at least once checked the loop.
- Generally do-while loop is osed for at least single ~~iterat~~ iterasive traversal.

Syntax
```
do {
   // code to be executed
} while (condition);
```
e.g.
```
#include <stdio.h>
int main()
{
   int i = 1 ; ①
   do <----------------------------------
   {                                      |
   printf ("%d. Jay Ganesha ... \n", i); ④ | B
   A  i++ ; ③                             |
   } while (i<=5) ; ② ---------------------
return 0;
}
```

Dry Run

Iteration I [i 2]

1: i = 1

// without checking Condition

4: 1. Jay Ganesha...

3: i++

Iteration II [i 2 3]

2: 2 <= 5 — True

4: 2. Jay Ganesha...

3: i++

Iteration III [i 3 4]

2: 3 <= 5 — True

4: 3. Jay Ganesha...

3: i++

Iteration IV

2: 4 <= 5 — True

4: 4. Jay Ganesha...

3: i++

Iteration V

2: 5 <= 5 — True

4: 5. Jay Ganesha...

3: i++

Iteration VI

2: 6 <= 5 — False

break.