ASSIGNMENT MODULE :- 3

1. Introduction to C++

Q-1. What are the key differences between Procedural Programming and Object-Oriented Programming (OOP)?

Ans.

1.Procedural Programming : Procedural programming is a programming paradigm that involves breaking down a program into a series of instructions, or procedures, that are executed in order.

2.Object-Oriented Programming (OOP) : Object-oriented programming (OOP) is a computer programming style that organizes software around objects, rather than logic and functions. In OOP, objects are units that contain both data and code, and programs are designed by making them out of objects that interact with each other.

Q - 2. List and explain the main advantages of OOP over POP.

Ans.

---> Security: OOP is more secure than POP because it supports access control and data hiding.

---> Code maintenance: OOP makes it easier to maintain and update code. OOP's modularity and encapsulation features allow coders to make changes without affecting the entire codebase.

---> Code reusability: OOP allows developers to reuse code blocks through inheritance.

---> Polymorphism: OOP's polymorphism concept allows objects from

different classes to be treated as instances of a common superclass. This enables different objects to respond differently to the same method invocation.

---> Inheritance: OOP supports inheritance, which allows developers to use attributes and functions of other classes.

---> Debugging: OOP makes debugging simpler.

---> Productivity: OOP practices can increase productivity.

---> Data management: OOP can streamline data management to reduce redundancy.

---> Code flexibility: OOP allows for adaptable solutions.

---> Problem-solving: OOP can help with problem-solving.

Q - 3. Explain the steps involved in setting up a C++ development environment.

Ans.

1.Install a Compiler: A compiler translates C++ code into machine code. Popular C++ compilers include.

(i). GCC -> Available for Linux, macOS, and Windows via MinGW.

(ii).MSVC (Microsoft Visual SC++): Comes with Visual Studio for Windows.

ex.

- **macOS:** Install Xcode or GCC via Homebrew (brew install gcc).
- **Linux:** Use your package manager (sudo apt install g++ for Ubuntu).
- **Windows:** Install MinGW or Visual Studio.

2. Install an IDE or Text Editor :

- **Visual Studio Code:** Lightweight and extensible.
- **Dev-C++:** Simple IDE for small projects.

3.Configure the Environment Variables (Optional) :

- Add the compiler (e.g., GCC) or build tool paths to the PATH environment variable.
- **Windows:** Use "System Properties > Environment Variables."

4. Verify the Setup

Test the setup by compiling a simple "Hello, World!" program:

```
#include<iostream>

using namespace std;

int main()

{

cout<<"\nHellow world!";

return 0;

}
```

Q - 4. What are the main input/output operations in C++? Provide examples.

Ans.

----> Main input/output in c++.

- **std : : cin :** for used to take a value by user.

- **std : : cout :** for used to print the value or string.

1. input using (std : : cin)

std : : cin is used to take input from the user.

ex.

```
#include<iostream>
using namespace std;
int main(){
int age;
cout<<"\nEnter the age = ";
cin>>age;

cout<<"\nUser enter the age is = ";

return 0;
}
```

2.Output using (std : : cout)

std::cout is used to display output to the console.

ex.

```
#include<iostream>
```

```
using namespace std;

int main(){

cout<<"\nThis is my program in c++";

return 0;

}
```

2. Variables, Data Types, and Operators

Q - 1. What are the different data types available in C++? Explain with examples.

Ans.

----> Two type in c++.

1. **primetive data types**
2. **non-primetive data type**

- drived data types
- user-defind data types

1. primetive data types

--- > these are basic data types built into thr language.

- integer(int) :

    --- > Used to store whole numbers.

        ex. int age = 25;

- floating point (float, double)

  --- > Used to store decimal numbers.

  ex. float marks = 98.88;

  ex.double largeValue = 12345.6789;

- character (char)

  --- > Used to store a single character.

  ex. char grade = 'A';

- boolan (bool)

  --- > Used to store true or false.

  ex. bool isvaild = true;

  ex. bool is vaild = false;

2. non-primetive data type

- **Derived datatype :** three type of derived datatypes

1. Array
2. Function
3. Pointer

- **Uerdefind datatype :** two types in Userdefind datatypes.

1. Structhar
2. Union

Q - 2. Explain the difference between implicit and explicit type conversion in C++.

Ans.

1.implicit type conversion :

----> Also known as type coercion, this is when the compiler automatically changes the data type of a value. This happens when the program or language calls for a different type than the one given by the programmer.

2.Explicit type conversion :

----> Also known as type casting, this is when the programmer intentionally changes the data type of a value. The programmer uses built-in functions or operators to do this.

Q - 3. What are the different types of operators in C++? Provide examples of each.

Ans.

1. **Arithmetic operators**
2. **Relational operators**
3. **Logical operators**
4. **Assignment operators**
5. **Bitwise operators**
6. **increment or decrement operators**
7. **ternry operators**

1. Arithmetic operator : used to basic mathemetic opretions addition, subtraction, maltiplication, division.

- **+ : addition**
- **- : subtraction**
- *** : maltiplication**
- **/ : division**

2. Relational operators : used to compering number or veriable, and returning true and false.

- **== : Both number are equal.**
- **< : one number are less a secound number.**
- **> : onr number are greter a secound number.**
- **!= : onr number are not equal a secound number.**
- **>= : one number are greter than equal a secound number.**
- **<= : one number are less than equal a secound number.**

3. Logical operators : used to combining two or more than conditions if they are true or falase.

- **&& : logocal And**
- **|| : logical Or**

4. Assignment operators : used to assingning value to veriable.

- **= : assign value**
- **+= : addition and assign value**
- **-= : subtract and assign value**

- ***= : maltiplication and assign value**
- **/= : division and assign value**

5. Bitwise operators : using to conditions compering shorts.

6. Increment or decrement operators : increment operator used to value increase and decrement operator used to value decrease.

- **a++ : post value increase.**
- **++a : pre value increase.**
- **a-- : post value decrease.**
- **--a : pre value decrease.**

7. ternary opreator : Used to evaluate a condition and return a value.

- ?: : condition ? a : b;

Q - 4. Explain the purpose and use of constants and literals in C++.

Ans. Constants and literals in C++ represent values that do not change during program execution. They improve code readability, maintainability, and help prevent accidental modification of values.

1.Constants : used to declared veriable to not changed the value.

ex. const int a = 10;

2. literals : used to declared veriable fixed values in the code

ex. int a = 10;

3. Control Flow Statements

Q - 1. What are conditional statements in C++? Explain the if-else and switch statements.

Ans. Condition statements in c++ control the flow of a program based on specific conditions. These conditions are usully boolean true or false. if the condition true a specific bloack of code are excueted, otherwise, another block is excuete.

----> The main conditional statement in c++

1. **if - else statement**
2. **switch statement**

1. **if - else statement**

i. simple if - else statement

ex. if (condition){

code } else {

code }

ii. ladder if - else statement

ex. if (condition){

code } else if (condition){

code }

iii. nested if - else statement

ex.if(condition){

if(condition){

```
code }
}
```

2. switch statement

```
ex. switch(expression)
{ case 1 :
  // code
  break;
  case 2 :
  // code
  break;
  defualt : // code
}
```

Q - 2. What is the difference between for, while, and do-while loops in C++?

Ans. Loops are used to in c++ to repeatedly execte a block of code as long as a specific condition is true.

----> The main types of loop in c++ are for, while , do while.

1. for loop : Used the number of iterations is known before hand or you want to control the loop.

ex. for(initialization,condition, increment/decrement)

{ //code }

2.while loop : Used the number of iterations is not known in advance and depends on a condition.

ex.while(condition)

{ // code }

3.do while loop : Used the loop must execute at least one regardless of the condition.

ex. do {

// code } while(condition)

Q - 3. How are break and continue statements used in loops? Provide examples.

Ans.

1. break statement : Break statement used to loop and switch statement.

The break statement is used to termineted the loop imedeatlly and exit for loop or execute next codition. and used to switch case statement in cheak to case and execute this code and condition are false to break the the code.

ex.#include<iostream>

using namespace std;

int main(){

int i,num = 10;

for(i = 1; i <= num; i++){

if(i == 5){

```
break;  }

}

cout<<"\nThe number =  "<< i ;

return 0;

}
```

2. countinue statement : Countinue statement used to skip the cuurent inertion of the loop and move to next inertion. used countionue statement is ignored the inertion.

```
ex.#include<iostream>

using namespace std;

int main(){

int i,num = 10;

for(i = 1; i <= num; i++){

if(i == 5){

countinue;  }

}

cout<<"\nThe number =  "<< i ;

return 0;

}
```

Q - 4. Explain nested control structures with an example.

Ans. Nested control structures like loops, conditional statements, or switches is placed inside another.

1.Nested if statement : an if satatement in other if or else block.

```
#include<iostream>
using namespace std;
int main()
{
   int num1,num2,num3;
cout<<"\nEnter the num1 value =  ";
cin>>num1;
cout<<"\nEnter the num2 value = ";
cin>>num2;
cout<<"\nEnter the num3 value = ";
cin>>num3;
   if(num1>num2)
   {

      if(num1>num3)
      {
         cout<<"\nnum1 is the biggest number =  "<<num1;
      }
      else
      {
```

```
        cout<<"\nnum3 is the biggest number = "<<num3;
      }
   }   else
{
      if(num2>num1)
      {
         if(num2>num3)
         {
            cout<<"\nnum2 is the biggest number = "<<num2;
         }
         else
         {
            cout<<"\nnum3 is biggest number = "<<num3;
         }
      }
   }
return 0;
}
```

2. Nested loop statement : an loop inside another loop.

ex.

```
#include<iostream>
```

```
using namespce std;
int main()
{
	int row,colm,i,j;
	cout<<"\nEnter the row = ";
	cin>>row;
	cout<<"\nEnter the colm  = ";
	cin>>colm;
	for(i=1;i<=row;i++)
	{
		for(j=1;j<=colm;j++)
		{
			cout<<"* ";
		}
		cout<<"\n";
	}
	return 0;
}
```

4. Functions and Scope

Q - 1. What is a function in C++? Explain the concept of function declaration, definition, and calling.

Ans. A function in C++ is a block of code that performs a specific task. and function is helpd to organizing code into smaller.

----> Four types in function.

1. with return type and with argument
2. With return type and without argument
3. without return type and with argument
4. without return type and without argument

----> Three types of key concepts :

- **declration of function :-** First step of Declar of and tell compiler about a function's existence before it is used in the program.
- **Calling :-** The function called and execute the code and function called always in main function.
- **definition :-** The function definition provides the actual implementation of the function.

Q -2. What is the scope of variables in C++? Differentiate between local and global scope.

Ans. The scope of a variable in C++ refers to the region or context within a program where the variable is declared and can be accessed.

1. **Local scope :-** variavle declared inside a function or block have local scope. They are acciblity with function or block.

   ex. { int x = 5; /* Local variable */ }

2. **Global scope :-** variable declared outside any funtion and class have global scope. they are acciblety throughout that fuction or block.

ex. int y = 10; /* Global variable */

Q - 3. Explain recursion in C++ with an example.

Ans. Recursion is a programming technique where a function calls itself to solve smaller instances of the same problem.

Ex. Factorial Calculation

The factorial of a number n, denoted as n!,is defind as :

$n!=n\times(n-1)\times(n-2)\times\ldots\times1$

Q - 4. What are function prototypes in C++? Why are they used?

Ans. A function prototype is a declaration of a function that specifies the function's name, return type, and parameters without providing its implementation. It acts as a blueprint or contract, informing the compiler about the function's existence and how it should be called.

Syntax of funcation :

return_type function_name(parameter_list);

- **return type:** Specifies the type of value the function will return.
- **function name:** The name of the function.
- **parameter list:** Specifies the types and optionally the names of parameters.

Function prototype Uesd :-

1. **Early Declaration :** A function prototype allows the compiler to know about a function before its actual implementation is encountered in the code. This is especially useful when functions are defined after the main() function.
2. **Type Checking :** The compiler uses the prototype to check that the

function is called with the correct number and types of arguments.

3. **Separation of Declaration and Definition :** Enables modular programming by separating the function's declaration (in header file) form its definition (in a Source file).

5. Arrays and Strings

Q - 1. What are arrays in C++? Explain the difference between single-dimensional and multi dimensional arrays.

Ans. The array can be defined as a group or collection of similar kinds of elements or data items that are stored together in contiguous memory spaces. All the memory locations are adjacent to each other, and the number of elements in an array is the size of the array.

1. **single-dimensional :** A One-Dimensional Array is a group of elements having the same data type which are stored in a linear arrangement under a single variable name.

2. **multi-dimensional arrays :** In a multi-dimensional array, each element in each dimension has the same, fixed size as the other elements in that dimension. In a jagged array, which is an array of arrays, each inner array can be of a different size. By only using the space that's needed for a given array, no space is wasted.

Q - 2. Explain string handling in C++ with examples.

Ans. The string class in C++ provides a more powerful and user-friendly way to handle strings. It offers various features such as dynamic sizing, inbuilt methods for manipulation, and easy concatenation.

Q - 3. How are arrays initialized in C++? Provide examples of both 1D and 2D arrays.

Ans. Arrays in C++ can be initialized in several ways, depending on their type and dimensions. Here's a breakdown of the initialization process for 1D arrays and 2D arrays:

1. One-Dimensional (1D) Arrays : 1D array is a linear collection of elements of the same type.

   ex. 1D array Declered by int arr1[5] = {1,2,3,4,5};

2. Two-Dimensional (2D) Arrays : A 2D array is a matrix-like collection of elements arranged in rows and columns.

   ex. 2D array Declered bt int arr1[2][3] = { {1,2,3} , {4,5,6}};

Q - 4. Explain string operations and functions in C++.

Ans. In C++, a string is a sequence of characters stored in a char array. C++ has a set of string functions and operations.
---> String can be defined using the string keyword or C-style string .

ex.string str = "Text" or char str[] = {'T', 'e' , 'x' , 't' , '\0'};

- **String functions**

---> Some common string functions in C++ include:

- **strlen(str1):** Finds the length of a string
- **strcpy(str1,str2):** Copies the content of one string to another
- **strcat(str1,str2):** Appends two strings
- **strcmp(str1,str2):** Compares two strings and returns a negative value if str1 is less than str2, 0 if they are equal, and a positive value if str1 is greater than str2

6. Introduction to Object-Oriented Programming

Q - 1. Explain the key concepts of Object-Oriented Programming (OOP).

Ans. Four key concepts in object-oriented programming language.

1. **Encapsulation**
2. **Abstraction**
3. **Inheritance**
4. **Polymorphism**

1. Encapsulation : Encapsulates all important information within an object, while only exposing select information. This protects the data from external interference.

2. Abstraction : Hides unnecessary implementation code, and only reveals internal mechanisms that are relevant to other objects. This allows developers to avoid repeating the same work multiple times.

3. Inheritance : Allows classes to reuse code and properties from other classes.

4. Polymorphism : Allows objects to share behaviors, and take on more than one form.

Q - 2. What are classes and objects in C++? Provide an example.

Ans. In C++, classes and objects are fundamental concepts of object-oriented programming.

1. **Classes**
2. **Object**

1. Classes : A class is a user-defined data type that serves as a blueprint for creating objects.

ex. class student

{ /* code */
};

2. Object : An object is an instance of a class. It contains the data (attributes) and can use the methods defined in the class.

ex. student s1;

Q - 3. What is inheritance in C++? Explain with an example.

Ans. Inheritance in C++ is a feature that allows a class to inherit properties and methods from another class, without modifying the original class.

----> some key concepts about inheritance in C++ :

1. **Base class :** The class that is being inherited from.
2. **Derived class :** The class that inherites from another class.
3. **Symbol for inheritance :** The **(:)** symbol is used to indicate inheritance.
4. **Modes of inheritance :** public, private, or protected can be used to specify how the inheritance is done.

Q - 4. What is encapsulation in C++? How is it achieved in classes?

Ans. In C++, encapsulation is the process of combining related data and functions into a single unit called a class.

---> To achieve encapsulation in C++ :

- **Declare class variables as private :** This prevents outside access to the data.
- **Provide public get and set methods :** This allows others to read or modify the value of a private member.