sjsytu

**Learning Report – Intermediate C++ and Testing**

Course Code: <CODE>

./



Table of Contents

[Activity – 1 - Intermediate C++ and Testing 3](#_Toc59870413)

[Goal……………………………………………………………………………………………………….3](#_Toc59870414)

[Type of Activity – Individual 3](#_Toc59870415)

[Set 1 - Classes and objects 3](#_Toc59870416)

[Set 2 - Operator overloading 3](#_Toc59870417)

[Set 3 – Inheritance 4](#_Toc59870418)

[Set 4 – Templates 4](#_Toc59870419)

[GitHub Link 4](#_Toc59870420)

[Difficulties 4](#_Toc59870421)

[Activity – 2 – Group Activity 5](#_Toc59870422)

[Goal………………………………………………………………………………………………………. 5](#_Toc59870423)

[Type of Activity – Group activity 5](#_Toc59870424)

1. Templates……………………….........................................................................................................5

[2. Additions of C++ from C 6](#_Toc59870426)

[3. Maps 6](#_Toc59870427)

[Activity – 3 – Individual Activity 7](#_Toc59870428)

[Goal………………………………………………………………………………………………………. 7](#_Toc59870429)

[Type of Activity – Individual activity 7](#_Toc59870430)

[References 8](#_Toc59870431)

# Activity – 1 - Intermediate C++ and Testing

## 

## **Goal**

To gain more knowledge in Intermediate C++, to have better understanding in testing the code and improving code quality according to industry standards.

## **Type of Activity – Individual**

### Set 1 - Classes and objects

A Class is a user defined data type, which holds data members and member functions, which can be accessed and used by creating object of that class.

The variables inside class definition are called as data members and the functions are called member functions.

Objects are instances of class, which holds the data variables declared in class and the member functions work on these class objects.

Each object has different data variables. Objects are initialized using special class functions called Constructors.

And whenever the object is out of its scope, another special class member function called Destructor is called, to release the memory reserved by the object.

### Set 2 - Operator overloading

It is a type of polymorphism in which an operator is overloaded to give user defined meaning to it. Overloaded operator is used to perform operations on user-defined datatype. For example '+' operator can be overloaded to perform addition on various data types, like for Integer, String (concatenation) etc.

Operator overloading can be done by implementing a function which can be:

1. Member Function
2. Non-Member Function
3. Friend Function

### Set 3 – Inheritance

 The class whose properties are inherited by other class is called the Parent class. And, the class which inherits properties of other class is called Child class.

Purpose of Inheritance

1. Code Reusability
2. Method Overriding (Hence, Runtime Polymorphism)

### Set 4 – Templates

The Standard Template Library (STL) is a set of C++ template classes to provide common programming data structures and functions such as lists, stacks, arrays, etc. It is a library of container classes, algorithms, and iterators. It is a generalized library and so, its components are parameterized. A working knowledge of template. Class is a prerequisite for working with STL.

**STL has four components**

* Algorithms
* Containers
* Functions
* Iterators

## **GitHub Link**

<https://github.com/STEPIN104305/Intermediate-CPP>

## 

## **Difficulties**

To write google test cases for a few programs in templates, and to write CanFrame program (Classes and Objects) but with the help of online resources I was able to successfully complete the task.

# 

# Activity – 2 – Group Activity

## **Goal**

To learn the concepts of templates, Additions of C++ from C and Maps.

## **Type of Activity – Group activity**

### 1.Templates:

Templates are powerful features of C++ which allows you to write generic programs. In simple terms, you can create a single function or a class to work with different data types using templates.

Templates are often used in larger codebase for the purpose of code reusability and flexibility of the programs.

**Types:**

* Function Templates
* Class Templates

**Function Templates:**

A function template works in a similar to a normal function, with one key difference is that a single function template can work with different data types at once but, a single normal function can only work with one set of data types.

However, a better approach would be to use function templates because you can perform the same task writing less and maintainable code.

**Syntax:**

template <class T>

T someFunction(T arg)

{

... .. ...

}

**Class Template:**

Like function templates, we can also create class templates for generic class operations. Sometimes, you need a class implementation that is same for all classes, only the data types used are different. Normally, you would need to create a different class for each data type or create different member variables and functions within a single class. However, class templates make it easy to reuse the same code for all data types.

**Syntax:**

template <class T>

class className

{

... .. ...

public:

T var;

T someOperation(T arg);

... .. ...

};

### 2. Additions of C++ from C:

* C++ is an object-oriented programming language where we can apply OOPS concepts.
* C++ is a base language for all the languages using OOPS concept.
* Exceptional Handling is available in C++ and not in C.
* In C++, for memory management we use new and delete operations.
* Access modifiers like private, public, protected is available in C++, not in C.
* Inheritance Feature is available in C++, not in C.
* C++ supports virtual functions and friend functions.
* Namespace feature is available in C++.

### 3. Maps:

Maps are associative containers that store elements in a mapped fashion. Each element has a key value and a mapped value. No two mapped values can have same key values.

Functions used in Map are as follows:

begin() – Returns an iterator to the first element in the map

end() – Returns an iterator to the theoretical element that follows last element in the map

size() – Returns the number of elements in the map

max\_size() – Returns the maximum number of elements that the map can hold

empty() – Returns whether the map is empty

pair insert(keyvalue, mapvalue) – Adds a new element to the map

erase(iterator position) – Removes the element at the position pointed by the iterator

erase(const g)– Removes the key value ‘g’ from the map

clear() – Removes all the elements from the map

# Activity – 3 – Individual Activity

## **Goal**

To have a hands on experience on STL concepts, string functions and test fixtures.

## **Type of Activity – Individual activity**

**Learning Outcomes**

* Used test fixtures to access functions without creating an object always and using the pointer to access the function in the class.
* Implemented various string operator overloading functions to implement operator overloading concepts.
* Learnt to implement various STL methods to perform operations and iterators to traverse within a list or vector or any type of STL.
* Codes executed are – MySring operations, Library\_Book using STL and test fixtures on account programs.

# 

# References

1. <https://www.geeksforgeeks.org/map-associative-containers-the-c-standard-template-library-stl/>
2. <https://www.geeksforgeeks.org/the-c-standard-template-library-stl/>
3. <https://www.learncpp.com/cpp-tutorial/template-classes/>
4. <https://www.learncpp.com/cpp-tutorial/introduction-to-operator-overloading/>