**Experiment No. 7**

**Title:** **Implementation of User Defined Templates**

**Batch: B2 Roll No.: 16010421119 Experiment No.: 7**

**Aim**: Write a C++ program to create a class ‘ArraySearch’ which has 1D array of generic type as an attribute. Define the following functions in the class:

input ( ) – to accept array elements from user

display( ) – to display array elements

find( ) – to find the smallest and largest elements from the array

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Resources needed: Text Editor, C++ compiler**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

### Theory:

**Template:**

Template is a concept which enables us to define generic classes and functions and thus provides support for generic programming. Generic programming is an approach where generic types are used as parameters in algorithms so that they work for a variety of suitable data types and data structures.

A template can be used to create a family of classes or functions. For example, a class template for an array class would enable us to create arrays of various data types such as int array and float array .Similarly, we can define a template for a function, say mul(),that would help us create versions of mul() for multiplying int, float and double type values.

In template when an object of a specific type is defined for actual use, the template definition for that class is substituted with the required data type. Since a template is defined with a parameter that would be replaced by a specified data type at the time of actual use of the class or function, the templates are sometimes called parameterized class or functions.

**Class Template:**

It is simple process to create a generic class using a template with an anonymous type. The general format of a class template is:

template<class T>

class classname

{

…

//class member specification

//with anonymous type T

//whenever appropriate

};

A class created from a class template is called a template class. The syntax for defining

an object of a template class is:

classname<type> objectname (arglist);

**Class Templates with Multiple Parameters:**

We can use more than one generic data type in a class. They are declared as a comma separated list within the template specification as shown below:

template<class T1, class T2, …..>

class classname

{

};

**Function Templates:**

Like class templates, we can also define function templates that could be used to create a family of functions with different argument types. The general format of a function template is:

template<class T>

returntype functionname (argument of type T)

{

//body of function

//with Type T

//whenever appropriate

}

**Function Template with Multiple Parameters:**

Like template classes, we can use more than one generic data type in the template statement, using a comma-separated list shown further:

template<class T1 , class T2, …..>

returntype functionname(arguments of types T1, T2, …)

{

}

**Member Function Templates:**

The member functions of the template classes themselves are parameterized by the type argument and therefore these functions must be defined by the function templates when these functions are defined outside the class. It takes the following general form:

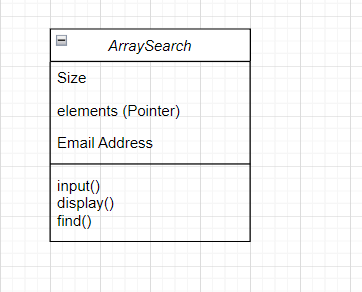
template <class T>

returntype classname <T> :: functionname(arglist)

{

}

**Class Diagram:**



\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Results: (Program with snapshot of output)**

**#include<iostream>**

**#include<iomanip>**

**using namespace std;**

**template<class A>**

**class ArraySearch**

**{**

**public:**

**A \*elements;**

**int size;**

**public:**

**ArraySearch(int s)**

**{**

**size = s;**

**elements = new A[size];**

**for(int i=0;i<size;i++)**

**{**

**elements[i]=0;**

**}**

**}**

**~ArraySearch()**

**{**

**delete elements;**

**}**

**void input()**

**{**

**for(int i=0;i<size;i++)**

**{**

**cout<<"Enter element: ";**

**cin>>elements[i];**

**}**

**}**

**void display()**

**{**

**for(int i=0;i<size;i++)**

**{**

**cout<<elements[i]<<endl;**

**}**

**}**

**void find()**

**{**

**int choice;**

**int max , min;**

**cout<<"Enter 1 to find largest number \n";**

**cout<<"Enter 2 to find smallest number \n";**

**cout<<"Enter choice: ";**

**cin>>choice;**

**if(choice==1)**

**{**

**max = elements[0];**

**for(int i = 0 ;i<size;i++)**

**{**

**if(elements[i]>max)**

**{**

**max = elements[i];**

**}**

**}**

**cout<<"Max number is: "<<max<<endl;**

**}**

**else**

**{**

**min = elements[0];**

**for(int j = 0 ;j<size;j++)**

**{**

**if(elements[j]<min)**

**{**

**elements[j] = min;**

**}**

**}**

**cout<<"Min number is: "<<min<<endl;**

**}**

**}**

**};**

**int main()**

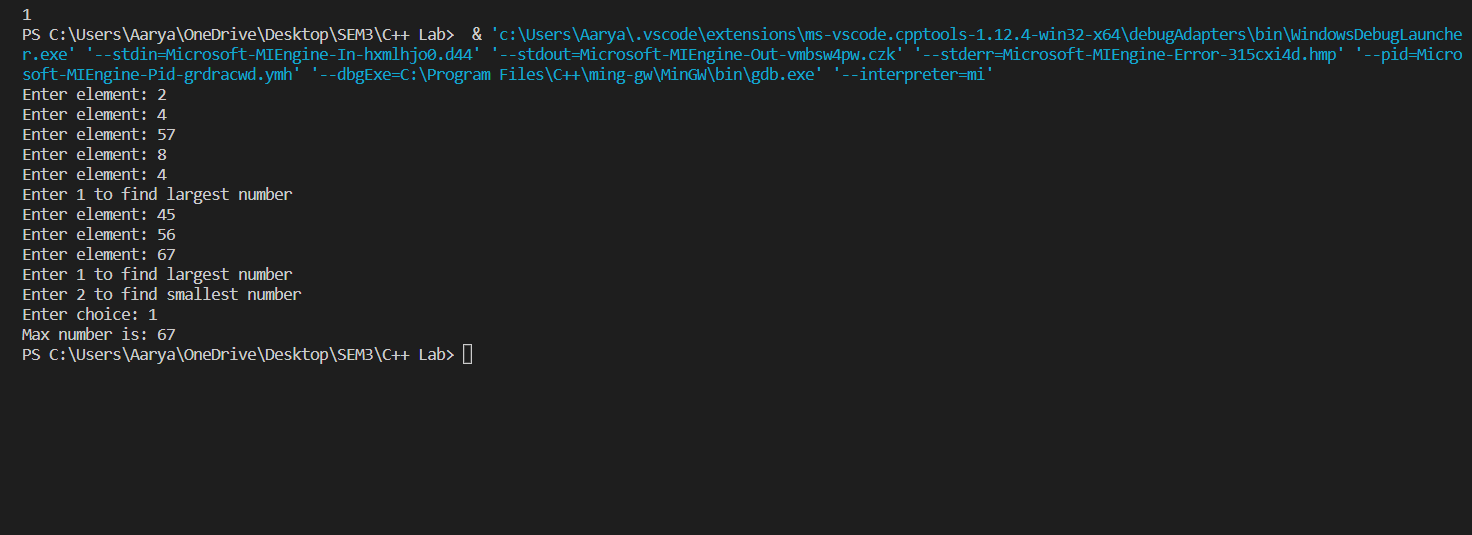
**{**

**ArraySearch<int>arr1(5);**

**arr1.input();**

**arr1.find();**

**}**

****

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Test Cases (minimum 5 test cases required):**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Sr. No.** | **Sample Input** | **Sample Output** | **Description** | **Test Case Type (general/special)** | **Pass/Fail** |
| **1** | **1** | **67** | **Maximum** | **General** | **Pass** |
| **2** | **2** | **23** | **Minimum** | **General** | **Pass** |
| **3** | **1** | **67** | **Maximum** | **General** | **Pass** |
| **4** | **2** | **23** | **Minimum** | **General** | **Pass** |
| **5** | **1** | **67** | **Maximum** | **General** | **Pass** |

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Questions:**

1. Compare template and macro.

1. Macro are used in C and C++ for replacement of numbers, small inline functions etc.

Template is only available in C++ language. It is used to write small macro like functions etc.

1. Macros are expanded by the preprocessor and then compilation takes place. Compiler will refer error messages in expanded macro or the line where macro has been called.
2. If parameters of any macro are given with operators ++ and -- with it and the macro used this argument more than one times then the increment or decrement of the original variable will that much times.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Outcomes:**

**CO3. Understand the concepts of streams and templates**

**Conclusion: (Conclusion to be based on the outcomes achieved)**

We can conclude that we have learnt about:-

1. Templates and inline functions
2. Difference in function templates and class templates.
3. Stream classes.

**Grade: AA / AB / BB / BC / CC / CD /DD**

Signature of faculty in-charge with date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**References:**

**Books/ Journals/ Websites:**

1. E Balagurusamy, Object oriented Programming with C++, Tata McGraw-Hill, 8th Edition September 2020
2. Herbert Schildt, C++: The Complete Reference, McGraw Hill Education, 4th edition, July 2017
3. Jeff Langr, Modern C++ Programming with Test-Driven Development : Code Better,Sleep Better, O′Reilly, 1st edition, November 2013
4. <https://docs.microsoft.com/en-us/cpp/cpp/?view=msvc-170>