OOP Lab Assignment 6

Jaynam Modi. PG-43. G3. Sep 15, 2020

- Aim : Create a class template stack.
- Problem Statement : Design a class Template to implement a stack consisting following member functions:
  - Create
  - Display
  - Push
  - Pop
- Objective: To study generic programming using class templates in C++.
- Theory:

A template is a simple and yet very powerful tool in C++. The simple idea is to pass data type as a parameter so that we don't need to write the same code for different data types. For example, a software company may need sort( ) for different data types. Rather than writing and maintaining the

multiple codes, we can write one sort() and pass data type as a parameter.

Function Templates We write a generic function that can be used for different data types. Examples of function templates are sort(), max(), min(), printArray().

Class Templates Like function templates, class templates are useful when a class defines something that is independent of the data type. Can be useful for classes like LinkedList, BinaryTree, Stack, Queue, Array etc.

Stacks are a type of container adaptors with LIFO(Last In First Out) type of working, where a new element is added at one end and (top) an element is removed from that end only.

- Algorithm :

1. Ask for stack inputs.
2. Push elements into stack.
3. The user inputs command to display stack.
4. The stack is displayed.

5. The user inputs command to delete element.

6. The last element is deleted.

- Input : User Choice 1- Push on stack, 2- Display stack, 3- Pop from the stack.
- Output : Stack contents.
- Platform/ Languages used : OS-64 bit Ubuntu 14.01, C++.
- Conclusion : Hence, learned class templates in C++ successfully.
- FAQs :

1. What are the advantages of templates in C++.

> C++ templates enable you to define a family of functions or classes that can operate on different types of information. Use templates in situations that result in duplication of the same code for multiple types. For example, you can use function templates to create a set of functions that apply the same algorithm to different data types.

2. Differentiate between function template and function overloading.

> Templates cannot take varying numbers of arguments. Overloads can. In addition, a template indicates that you can operate on any data type, but it's pointless to represent this when in fact, the vast, vast majority of templates would be specializations only (in your system). Also, overloads can be virtual and template specializations cannot. Nor can specializations differ in their signatures from the base.