**Assignment number: 10**

**Subject: Object Oriented Programming**

**TEMPLATE**

Name: ***RIA MITTAL***

Class: ***SECOND YEAR ENGINEERING***

Division: ***B***

Roll no: ***222008***

Batch: ***B1***

**Problem Statement:**

Write a function template selection Sort. Write a program that inputs, sorts and outputs an integer array and a float array.

# Objective :

## **Assume appropriate data members and member function to accept required data & print data .**

# CONCEPT:

Template function

# THEORY:

**Class**

**Templates** are a feature of the [C++](https://en.wikipedia.org/wiki/C%2B%2B) programming language that allows functions and classes to operate with [generic types](https://en.wikipedia.org/wiki/Generic_programming). This allows a function or class to work on many different [data types](https://en.wikipedia.org/wiki/Datatype) without being rewritten for each one.

Templates are of great utility to programmers in C++, especially when combined with [multiple inheritance](https://en.wikipedia.org/wiki/Multiple_inheritance) and [operator overloading](https://en.wikipedia.org/wiki/Operator_overloading). The [C++ Standard Library](https://en.wikipedia.org/wiki/C%2B%2B_Standard_Library) provides many useful functions within a framework of connected templates.

Major inspirations for C++ templates were the parameterized modules provided by [CLU](https://en.wikipedia.org/wiki/CLU_(programming_language)).

A *function template* behaves like a function except that the template can have arguments of many different types (see example). In other words, a function template represents a family of functions. The format for declaring function templates with type parameters is:

template <class identifier> function\_declaration;

template <typename identifier> function\_declaration;

Both expressions have the same meaning and behave in exactly the same way. The latter form was introduced to avoid confusion,[[2]](https://en.wikipedia.org/wiki/Template_(C%2B%2B)" \l "cite_note-2) since a type parameter need not be a class. (it can also be a basic type such as int or double.)

For example, the C++ Standard Library contains the function template max(x, y) which returns the larger of x and y. That function template could be defined like this:

-------------------------------------------------------------------------------------------------

template <type name T>

inline T max(T a, T b)

{

return a > b ? a : b;

}

-------------------------------------------------------------------------------------------------

This single function definition works with many data types. The usage of a function template saves space in the source code file in addition to limiting changes to one function description and making the code easier to read.

A template does not produce smaller object code, though, compared to writing separate functions for all the different data types used in a specific program. For example, if a program uses both an int and a double version of the max() function template shown above, the compiler will create an object code version of max() that operates on int arguments and another object code version that operates on double arguments. The compiler output will be identical to what would have been produced if the source code had contained two separate non-templated versions of max(), one written to handle int and one written to handle double.

**Switch-case**

A switch statement allows a variable to be tested for equality against a list of values. Each value is called a case, and the variable being switched on is checked for each case.

The default case isn't always required and isn't compulsory.

-------------------------------------------------------------------------------------------------------------

**Syntax:**

The syntax for a switch statement in C++ is as follows:

switch(expression){

case constant-expression :

statement(s);

break;

case constant-expression :

statement(s);

break;

default :

statement(s);

}

**ALGORITHM**

1. Start.

2. Initialize the variables a, T in template class .

3. Create a template class <T>

4. Declare and define functions getdata(), display().

5. Create object of class T in the main function.

6. Display the sorting menu.

7. Call the functions of class accordingly.

8. Stop.

**STATE TRANSITION DIAGRAM**

Q0=Start.

Q1=initialize variables and ask for arguments to be passed.

Q2=Pass argument to sort() function using switch case.

Q3=Call sort() function.

Q4=Display sorted array.

Qe=Stop.

**PROGRAM:**

#include<iostream>

using namespace std;

template<class T>

class Sorting

{

public:

T sort();

};

template<class T>

T Sorting <T>::sort()

{

T size;

T arr[100];

T temp;

int i,j;

cout<<"\nenter the size of array\n";

cin>>size;

cout<<"\nenter the array elements\n";

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

{

cin>>arr[i];

}

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

{

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

{

if(arr[i]<arr[j])

{

temp=arr[j];

arr[j]=arr[i];

arr[i]=temp;

}

}

}

cout<<endl;

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

{

cout<<arr[i]<<endl;

}

}

int main()

{

Sorting<int> obj;

cout<<"\ninteger array sorting\n";

obj.sort();

Sorting<float> ob;

cout<<"\nfloat array sorting\n";

ob.sort();

return 0;

}

**OUTPUT:**

integer array sorting

enter the size of array

5

enter the array elements

2

1

33

44

22

1

2

22

33

44

float array sorting

enter the size of array

6

enter the array elements

12.2

22.1

55.5

44.4

33.1

8.8

8.8

12.2

22.1

33.1

44.4

55.5

# CONCLUSION:

Through this program we understand the implementation of Template Function and function calling , and the use of switch-case as a conditional statement in place of if-else-if compound statement.

---------------------------------------------------------------------------------------------------------