G. H. Raisoni College Of Engineering And Management, Wagholi Pune

2021- 2022

Assignment no :- 9

Department	CE [SUMMER 2022 (Online)]		
Term / Section	III/B	Date Of **submission**	10-12-2021
Subject Name /Code	Object Oriented Programming/ UTIL201/UITP201		
Roll No.	SCOB77	Name	Pratham Rajkumar pitty
Registration Number	2020AC0E1100107		

Assignment No 9

Aim :→ Write a function template for finding the minimum value contained in an array

Theory :→

Function templates are special functions that can operate with generic types. This allows us to create a function template whose functionality can be adapted to more than one type or class without repeating the entire code for each type.

In C++ this can be achieved using template parameters. A template parameter is a special kind of parameter that can be used to pass a type as argument: Just like regular function parameters can be used used to pass ~~a type as~~ values to a function, template parameters allow to pass also types to a function. These function templetes can use these parameters as if they were any other regular type.

The format for declaring function templates with type parameters is:

```
template < class identifier> function_declaration;
template < typename identifier> function_declaration;
```

The only difference between both prototypes is the use of either the keyword class or the keyword typename. Its use is indistinct, since both expressions have exactly the same meaning and behave exactly the same way.

Ex

Create a template function that returns the greater one of two objects we could use

```
template < class myType >
myType GetMax (myType a, myType b)
{ return (a>b?a:b); }
```

••• C++ Template

Templates are powerful features of C++ which allows you to write generic programs.

OR

We 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.

~~The function~~

- The concept of templates can be used in two different ways:

- Function Templates
- Class Templates

•• Function Templates

A function template starts with the keyword template followed by template parameter(s) inside <>

Which is followed by the function defination.

```
template <typename T>
T functionName(T parameter1, T parameter2,
 {
     // code
 }
```

In the above code, T is a template argument that accepts different data types (int, float, etc.) and typename is a keyword.

When an argument of a data type is passed to functionName() the compiler generates a new version of functionName() for a given data type

Calling a Function Template

After declaration & defination of a function template. we call it in other functions or templates such as the main() function) with the following syntax

FunctionName<dataType>(parameter1, parameter2, ...);

Ex

A templete that adds two numbers:

```
template<typename T>
T add(T num1, T num2)
{ return (num1+num2);}
// we can then call in main() function to add
// int and double numbers.
int main(){
int result1;
```

```
double result2;
// calling with int parameters
result = add<int>(2,3);
cout << result1 << endl;

Calling with double parameters
result2 = add<double>(2.2 , 3.3);
cout << result2 << end;

return 0;
}
```

Class Template Declaration

Class Template is used to ~~write~~ ~~generic~~ ~~program~~ create a single class to work with different datatypes

A class template starts with the keyword template followed by template parameter(s) inside <>

```
template <class T>
class className {
  Private :
  T var;
  ... ... ...
  Public :
  T functionName (T arg);
  ... ... ...
};
```

T → is the template argument which is a placeholder for the data type used, and

Class → is a keyword.

• • Creating a Class Template Object

Syntax: ClassName <dataType> Class Object;

EX

ClassName<int> Class object;

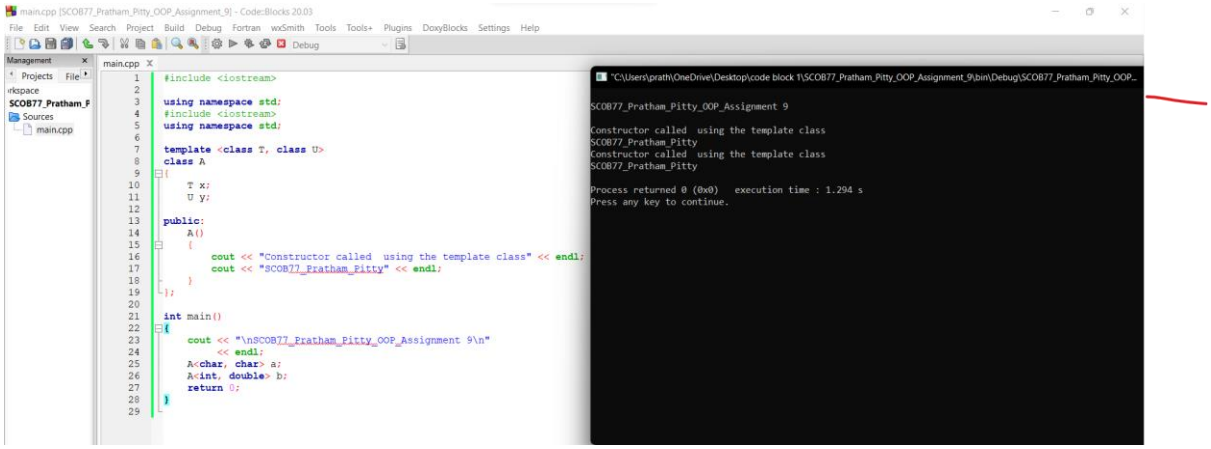
we can also ~~de~~ do.

(1) Define a class member outside the class Template
(2) Class template with multiple parameters

Program code

```
#include <iostream>


using namespace std;
#include <iostream>
using namespace std;


template <class T, class U>
class A
{
   T x;
   U y;


public:
   A()
   {
      cout << "Constructor called  using the template class" << endl;
      cout << "SCOB77_Pratham_Pitty" << endl;
   }
};


int main()
{
   cout << "\nSCOB77_Pratham_Pitty_OOP_Assignment 9\n"
       << endl;
   A<char, char> a;
   A<int, double> b;
   return 0;
}
```

Output:-


main.cpp [SCOB77_Pratham_Pitty_OOP_Assignment_9] - Code::Blocks 20.03
File Edit View Search Project Build Debug Fortran wxSmith Tools Tools+ Plugins DoxyBlocks Settings Help
Debug
Management
Projects
Sources
main.cpp
#include <iostream>
using namespace std;
#include <iostream>
using namespace std;
template <class T, class U>
class A
{
T x;
U y;
public:
A()
{
cout << "Constructor called using the template class" << endl;
cout << "SCOB77_Pratham_Pitty" << endl;
}
};
int main()
{
cout << "\nSCOB77_Pratham_Pitty_OOP_Assignment 9\n"
<< endl;
A<char, char> a;
A<int, double> b;
return 0;
}
"C:\Users\prath\OneDrive\Desktop\code block 1\SCOB77_Pratham_Pitty_OOP_Assignment_9\bin\Debug\SCOB77_Pratham_Pitty_OOP_
SCOB77_Pratham_Pitty_OOP_Assignment 9
Constructor called using the template class
SCOB77_Pratham_Pitty
Constructor called using the template class
SCOB77_Pratham_Pitty
Process returned 0 (0x0) execution time : 1.294 s
Press any key to continue.