VYSYA COLLEGE, SALEM-103

CLASS: I BCA **PROGRAMS – 14 TO 16**

SUBJECT: PRACTICAL: C++ PROGRAMMING **SUBJECT CODE: 22UCAP02**

EX. NO: 14 – PROGRAM USING CLASS TEMPLATE.

AIM:

To write a C++ program to Demonstrate Class Template.

PROCEDURE:

Step 1 : Start the program

Step 2 : Include necessary header files

Step 3 : Include standard namespace

Step 4 : Define the template class Number

Step 5 : Implement the main function

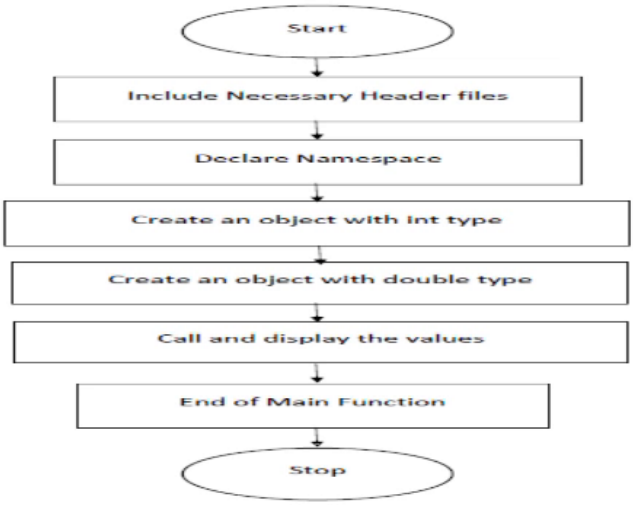
Step 6 : Compile and execute to Display the values

Step 7 : Return statement

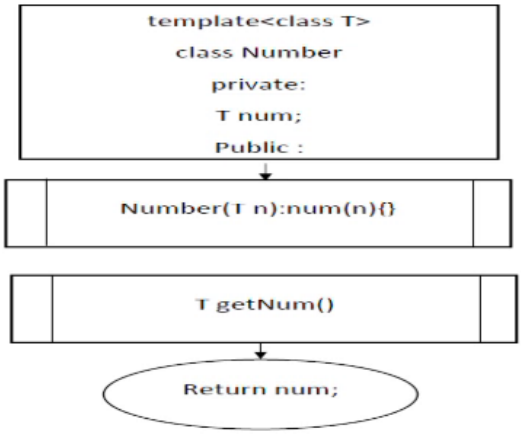
Step 8 : Stop the program

FLOWCHART :

Main program:



Template



SOURCE CODE :

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

// Class template
template<class T>
class Number
{
private:
   // Variable of type T
   T num;

public:
Number(T n) : num(n) {}                // constructor

   T getNum() {
return num;
   }
};

int main() {

   // create object with int type
   Number<int>numberInt(7);

   // create object with double type
   Number<double>numberDouble(7.7);
```

```
cout<<"int Number = "<<numberInt.getNum() <<endl;
cout<<"double Number = "<<numberDouble.getNum() <<endl;

return 0;
}
```

Output:

```
int Number = 7
double Number = 7.7
```

RESULT:

Thus, the Demonstrate Class Template program was executed successfully.

EX. NO: 15 - PROGRAM USING FUNCTION TEMPLATE

AIM :

To write a C++ program to Demonstrate Function Template.

PROCEDURE :

Step 1 : Start the program

Step 2 : Include necessary header files

Step 3 : Define the function template add

Step 4 : Implement the main function

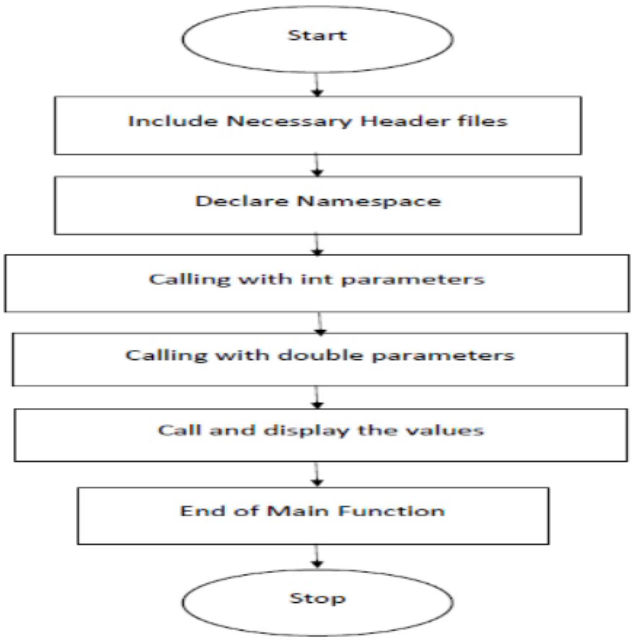
Step 5 : Compile and execute to Display the values

Step 6 : Return statement

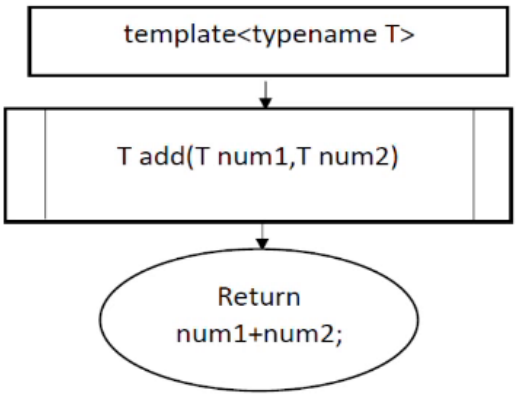
Step 7 : Stop the program

FLOWCHART :

MAIN PROGRAM:



Function



SOURCE CODE :

```
#include <iostream>
using namespace std;
template<typename T>
T add(T num1, T num2)
{
    return (num1 + num2);
}
int main()
{
    int result1;
    double result2;
    // calling with int parameters
    result1 = add<int>(2, 3);
    cout<<"2 + 3 = "<< result1 <<endl;
    // calling with double parameters
    result2 = add<double>(2.2, 3.3);
    cout<<"2.2 + 3.3 = "<< result2 <<endl;
    return 0;
}
```

OUTPUT:

```
2 + 3 = 5
2.2 + 3.3 = 5.5
```

RESULT:

Thus, the Demonstrate Function Template program was executed successfully

Ex. No : 16 –PROGRAM USING EXCEPTION HANDLING

Aim :

To write Program using the concept of exception handling.

Procedure :

Step 1 : Start the program

Step 2 : Include necessary header files

Step 3 : Declare variables to store Numerator , Denominator and result

Step 4 : Implement the main function

Step 5 : Assign the values for the input variables .

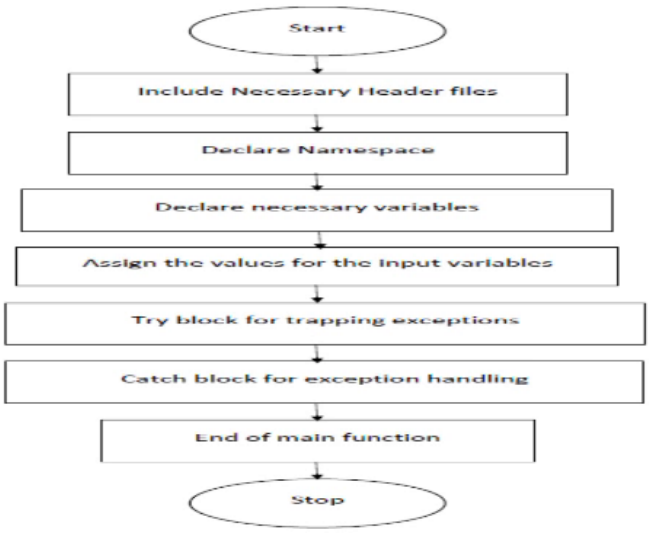
Step 6 : try block with exception handling

Step 7 : Catch block for exception handling

Step 8 : Return statement

Step 9 : Stop the program

FLOWCHART



SOURCE CODE :

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

int main() {
double numerator, denominator, divide;
cout<<"Enter numerator: ";
cin>> numerator;

cout<<"Enter denominator: ";
cin>> denominator;

try {
// throw an exception if denominator is 0
if (denominator == 0)
throw 0;

// not executed if denominator is 0
divide = numerator / denominator;
cout<< numerator <<" / "<< denominator <<" = "<< divide <<endl;
   }

catch (intnum_exception) {
cout<<"Error: Cannot divide by "<<num_exception<<endl;
   }

return 0;
}
```

OUTPUT:

Enter numerator: 123

Enter denominator: 4

123 / 4 = 30.75

OUTPUT 2:

Enter numerator: 333

Enter denominator: 0

ERROR!

Error: Cannot divide by 0

RESULT:

Thus, the Demonstrate of exception handling was executed successfully.

.