LAB SHEET - 11

[Template and Exception Handling]

Date: 2023/01/26

#*Index Page:-*

#Topics: #Page No.

1. Title:

Template and Exception Handling

2. Objective:

- To be familiar with class template and function template.
- To be familiar with exception handing mechanism and implement them wherever necessary.

3. Theory:

- ***Introduction to generic programming.***

  Generic programming is about **generalizing software components so that they can be easily reused in a wide variety of situations**. In C++, class and function templates are particularly effective mechanisms for generic programming because they make the generalization possible without sacrificing efficiency.

- ***Function template with their syntax.***

  A template is **a simple 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 to sort() for different data types.

  Syntax:

```
template<class T>
void TT(TT x,TT y){
        cout<<"……..."<<(x+y)<<endl;
}
```

- ***Class template with their syntax.***

  A class template **provides a specification for generating classes based on parameters**. Class templates are generally used to implement containers. A class template is instantiated by passing a given set of types to it as template arguments.

  Syntax:

```
template<class type>
ret-type func-name(parameter list) {
 //body of the function
}
```

- ***Exception handling mechanism in C++.***

  Exception handling is **a mechanism that separates code that detects and handles exceptional circumstances from the rest of your program**. Note that an exceptional circumstance is not necessarily an error. When a function detects an exceptional situation, you represent this with an object.

4. Lab Exercise:

a. *Create a function templates to find the sum of two integers and floats using function template.*

#SourceCode:

```
#include<iostream>
using namespace std;
template <class SS>
void sum(SS x,SS y){
   cout<<"Sum of two numbers: "<<(x+y)<<endl;
}
int main(){
   sum(7,9);
   sum(424.353f,352.642f);
   return 0;
}
```

#Output:

```
Largest data among two datas: 9
Largest data among two datas: 424.353
Largest data among two datas: r

Process returned 0 (0x0)   execution time : 1.017 s
Press any key to continue.
```

b. *Write a C++ Program to find Largest among two integers, characters and float using function template.*

#SourceCode:

```
#include<iostream>
using namespace std;
template <class L>
void LN(L x,L y){
   if(x>y){
      cout<<"Largest data among two datas: "<<x<<endl;
   }
   else{
      cout<<"Largest data among two datas: "<<y<<endl;
   }
}
int main(){
   LN(7,9);
   LN(424.353f,352.642f);
   LN('r','e');
   return 0;
}
```

#Output:

```
Sum of two numbers: 16
Sum of two numbers: 776.995

Process returned 0 (0x0)   execution time : 1.109 s
Press any key to continue.
```

c. Create a function template to swap two values.

#SourceCode:

```
#include<iostream>
using namespace std;
template <class SP>
void swaap(SP &x,SP &y){
   cout<<"Given two values: "<<x<<","<<y<<endl;
   int z=x;
   x=y;
   y=z;
   cout<<"Swaped Values: "<<x<<","<<y<<endl;
}
int main(){
   int a=12910;
   int b=47747;
   swaap(a,b);
   return 0;
}
```

#Output:

```
Given two values: 12910,47747
Swaped Values: 47747,12910

Process returned 0 (0x0)   execution time : 0.875 s
Press any key to continue.
```

d. WAP to find the roots of quadratic equation using function template.

#SourceCode:

```
#include<iostream>
#include<math.h>
using namespace std;
template <class EQ>
void Qeqn(EQ a,EQ b,EQ c){
   EQ d;
   d=((b*b)-(4*a*c));
```

```
    if(d>0){
        EQ e=-b+sqrt(d);
        e=e/(2*a);
        cout<<"Roots are unequal and real."<<endl;
        cout<<"Root a: "<<e<<endl;
        e=-b-sqrt(d);
        e=e/(2*a);
        cout<<"Root b: "<<e<<endl;
    }
    else if(d<0){
        cout<<"Roots are unequal and imaginary."<<endl;
    }
    else{
        cout<<"Roots are equal and same."<<endl;
        EQ e=-b+sqrt(d);
        e=e/(2*a);
        cout<<"Root a: "<<e<<"Root b: "<<e<<endl;
    }
}
int main(){
    Qeqn(4,7,2);
    int x,y,z;
    cout<<"\nEnter any three numbers: "<<endl;
    cin>>x;
    cin>>y;
    cin>>z;
    Qeqn(x,y,z);
    return 0;
}
```

#Output:

```
Roots are unequal and real.
Root a: 0
Root b: -1

Enter any three numbers:
2
4
8
Roots are unequal and imaginary.

Process returned 0 (0x0)   execution time : 7.671 s
Press any key to continue.
```

e. *WAP to perform sum and product of two integer and two floating point data using class template.*

#SourceCode:

```
#include<iostream>
using namespace std;
template <class data>
void sumpro(data x,data y){
   cout<<"Sum of two numbers: "<<(x+y)<<endl;
   cout<<"Product of two numbers: "<<(x*y)<<endl;
}
int main(){
   cout<<"Two integer numbers: "<<endl;
   sumpro(44,66);
   cout<<"\nTwo float numbers: "<<endl;
   sumpro(33.222f,55.111f);
   return 0;
}
```

#Output:

```
Two integer numbers:
Sum of two numbers: 110
Product of two numbers: 2904

Two float numbers:
Sum of two numbers: 88.333
Product of two numbers: 1830.9

Process returned 0 (0x0)   execution time : 0.953 s
Press any key to continue.
```

f. *Create a class template to find the scalar product of vectors of integers and vectors of floating point number.*

#SourceCode:

```
#include<iostream>
using namespace std;
template <class Z=float>
class vvector{
private:
   Z xco,yco;
public:
   vvector(Z xco=0,Z yco=0){
      this->xco=xco;
      this->yco=yco;
   }
```

```
    void ScalarPro(vvector V1,vvector V2){
        xco=V1.xco*V2.xco;
        yco=V1.yco*V2.yco;
    }
    void display(){
        cout<<"X Co-ordinate: "<<xco<<", Y Co-ordinate: "<<yco<<endl;
    }
};
int main(){
    vvector <> V1(4.3,5.5);
    vvector <> V2(3.2,1.2);
    vvector <> V3;
    V3.ScalarPro(V1,V2);
    V3.display();
    return 0;
}
```

#Output:

```
X Co-ordinate: 13.76, Y Co-ordinate: 6.6

Process returned 0 (0x0)   execution time : 0.921 s
Press any key to continue.
```

g. Write a program to illustrate exception handling mechanism in C++.

#SourceCode:

```
#include<iostream>
using namespace std;
int main(){
    float a,b;
    cout<<"Enter value of a: ";
    cin>>a;
    cout<<"Enter value of b: ";
    cin>>b;
    try{
        if(b==0){
            throw b;
        }
        else{
            cout<<"Value: "<<(a/b)<<endl;
        }
    }
    catch(float b){
        cout<<"Cannot divide any integer by zero."<<endl;
    }
```

```
    return 0;
}
```

#Output:

```
Enter value of a: 4
Enter value of b: 8
Value: 0.5

Process returned 0 (0x0)   execution time : 5.165 s
Press any key to continue.
```

h. Write program that catches multiple exceptions.

#SourceCode:

```
#include<iostream>
using namespace std;
void catches(int x){
    try{
        if(x==0||x==1){
            throw x;
        }
        else if(x!=0&&x!=1){
            throw 'x';
        }
    }
    catch(int y){
        cout<<"Caught a boolean"<<endl;
    }
    catch(const char z){
        cout<<"Caught a integer"<<endl;
    }
}
int main(){
    cout<<"1st: "<<endl;
    catches(6);
    cout<<"2nd: "<<endl;
    catches(0);
    cout<<"3rd: "<<endl;
    catches(4);
    return 0;
}
```

#Output:

```
1st:
Caught a integer
2nd:
Caught a boolean
3rd:
Caught a integer

Process returned 0 (0x0)   execution time : 1.309 s
Press any key to continue.
```

5. Conclusion:

In this LabSheet, we become familiar with class template, function template and exception handing mechanism and we also learned how and where to implement it.