Assignmen 3

CAP : 444

Course Title: C++ Theory

Name : Jayshri Lal Pandit

Roll No. : RD2112A103

Reg No. : 12111670

(1.) Explain the exception handling mechanism in detail. Discuss the multiple catch statements with the help of program.

Ans:- Exception is an event that disturbs the normal flow of the program and that why result out of expectation.

C++ provide a way to transfer control from one part of a program to another that is called exception handling in C++. Exception handling is built upon three keywords : try, catch and throw.

- try :- The try block contain statements which may generate exceptions. It is followed by one or more catch blocks.
- throw :- When an exception occurs in try block, it is thrown to the catch block using throw keyword. This is done by using a throw keyword.
- catch :- The catch block defines the action to be taken, when an exception occurs. The catch keyword indicates the catching of an exception.

P.T.O

Syntax :-

```
try
{
  // Block of code of try

  throw exception ; /* Throw an exception
  when a problem arise. */

}
catch ( )
{

  // Block of code to handle exception

}
```

C++ exception hadling is built upon three keywords: try, catch and throw
- throw A program throws an exception when a problem shows up. This is done using a throw keyword. catch keyword catches an exception with an exception handler at the place in a program where you want to handle the problem.

/* Demo program which demonastrate how to use multiple catch statement in exception handling

Program */

```
#include <iostream>
using namespace std;
int main ( )
{
    int choice;

  try
  {
    cout<<" Enter any choice: ";
    cin >> choice;

    if (choice ==0 )
        cout<<" Hello sir! "<<endl;
    else if ( choice = = 1)
        throw (100); // throw integer value
    else if ( choice == 2)
        throw ('x'); // throw integer value

    else if (choice ==3)
        throw (1.23f); // throw float value.
    else
        cout<<" Bye sir! "<<endl;
  }
  catch (int a)
  {
    cout <<" Integer Exception Block, value is: "
                                <<0<<endl;
  }
```

```
catch (char b)
{
   cout<<" character Exception Block, value is:
                                    <<b<<endl;
}
catch(float c)
{
   cout<<" Float Exception Block, value is:
                                    <<c<<endl;
}
 return 0;

}
```

Output:

First Run:
Enter any choice: 0 ↵ enter
Hello sir!

Sencon Run:
Enter any choice: 1 ↵ enter
Integer Exception Block, Value is: 100

Third Run:
Enter any choice: 2
Character Exception Block, Value is: X

Fourth Run:
Enter any choice: 3
Float Exception Block, Value is: 1.23

Fifth Run:
Enter choice: 4
Bye sir!

(2.) Write a program to achieve the following:

a.) implement class template
b.) implement swap operation using function template.

Ans:- (a.)

```
/* Demo program to illustrate the
   implementation of class template*/

#include<iostream>
Using namespace std;

template < class T >
class calculator
{
   private:
      T num1, num2;
   public:
      calculator (T n1, T n2)
      {
         num1 = n1;
         num2 = n2;
      }
```

P.T.O

```
void displayResult()
{
  cout<< "Numbers: " << num1 << "and" << num2
                << "." << endl;
  cout << num1 << " + " << num2 << " = "
                << add() << endl;

  cout << num1 << " - " << num2 << " = "
                << subtract() << endl;
  cout << num1 << " * " << num2 << " = "
                << multiply() << endl;

  cout << num1 << " / " << num2 << " = "

                << divide() << endl;
}

T add()
{
    return num1 + num2;
}

T subtract()
{
  return num1 - num2;
}
T multiply()
{
  return num1 * num2;
}
```

```
T divide()
{
    return num1 / num2;

  }
};

int main()

{
  Calculator<int> intCalc(2, 1);
  Calculator<float> floatCalc(2.4, 1.2);

  cout << " Int results : " << endl;

   intCalc.displayResult();

  cout << endl << " Float results : " << endl;

   floatCalc.displayResult();

   return 0;
}
```

<Output>

```
Int results:
Numbers : 2 and 1.
2+1 = 3
2-1 = 1
2*1 = 2
2/1 = 2
```

Float results:
Numbers: 2.4 and 1.2.
2.4 + 1.2 = 3.6
2.4 - 1.2 = 1.2
2.4 * 1.2 = 2.88
2.4 / 1.2 = 2

(b) /* Demo program to implement Swap operation using function template */

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

template <class T>

Void Swap (T &X, T &Y)
{
    T temp;

    temp = X;
     X   = Y;
     Y   = temp;

}
int main ()
{
   int x, y;
  cout<<" Enter two numbers:";
  cin >> X >> Y;
```

```
cout << " Before swap : ";

cout << "\n x value is : " << x;
cout << "\n y value is : " << y;

Swap ( x, y );

cout << "\n\n After function Templates : \n";

cout << "\n value is : " << x;
cout << "\n value is : " << y;

return 0;
}
```

OUTPUT

```
Enter two value numbers : 10 20  <- enter
Befor Swap :
 x value is : 10
 y value is : 20

After Function Templates :
 x value is : 20
 y value is : 10
```

(3.) Discuss recursion with template function and difference between templates and macros.

Ans:→ We use a recursive template function to iterate over all the parameters one by one. A template function that allows an arbitrary number of arguments of a few different types which will be processed one by one sequentially.

```
/* Demo program to illustrate use
   of template function to recursively
   calculate the sum of the first n
   elements */

#include <iostream>
using namespace std;

template <class T>
T Rsum(T a[], int n)
{
    if (n<=0)
    {
        return 0;
    }
    else
    {
        return Rsum(a, n-1) + a[n-1];
    }
int main()
{
```

```
float [redacted] a[] = {1,2,3,0,6,5,4};

cout << Rsum(a,-2) << endl;
cout << Rsum(a,5) << endl;
cout << Rsum(a,6) << endl;

return 0;

}
```

Output

```
0
12
17
```

Deffirence between macro and function:-

Macro	Templates
(1.) Macro is preprocessed.	(1.) Template is compiled.
(2) Macro can be argumented.	(2.) Template is also can be argumented
(3) In macro we can pass arguments one type.	(3) In template we can pass arguments different type.

P.T.O

(4.) Before compilation, macro name is replaced by macro value.	(4.) During template call, tranfer of control takes place.
(5.) Speed of execution using macro is faster.	(5) Speed of execution using template is slower.
(6) Macros are useful when small code is repeated many times	(6) Template are useful when large code is to be written as genaralised.
(7) Using macro increases the code length.	(7.) Using template keeps the code length unaffacted.

The End