Subject Name: **Object Oriented Programming Using C++**

Subject Code: **BCA-301 N**

Subject Topic: **Exception Handling**

Abhishek Dwivedi

Assistant Professor

Department of Computer Application

UIET, CSJM University, Kanpur

Exception Handling in C++

- Errors can be broadly categorized into two types. We will discuss them one by one.

a. Compile Time Errors

b. Run Time Errors

- **Compile Time Errors** – Errors caught during compiled time is called Compile time errors. Compile time errors include library reference, syntax error or incorrect class import.
- **Run Time Errors -** They are also known as exceptions. An exception caught during run time creates serious issues.

Run Time Error

- Exception handling is the process of handling errors and exceptions in such a way that they do not hinder normal execution of the system. For example, User divides a number by zero, this will compile successfully but an exception or run time error will occur due to which our applications will be crashed.
- In C++, Exception handling is done using three keywords:

a. try
b. catch
c. throw

- **Syntax:**

```
try
{
   //code throw parameter;
}
catch(exception_name  ex)
 {
   //code to handle exception
 }
```

- **try** block:- The code which can throw any exception is kept inside(or enclosed in) a try block. Then, when the code will lead to any error, that error/exception will get caught inside the catch block.
- **catch** block:- catch block is intended to catch the error and handle the exception condition. We can have multiple catch blocks to handle different types of exception and perform different actions when the exceptions occur. For example, we can display descriptive messages to explain why any particular exception occurred.
- **throw** statement:- It is used to throw exceptions to exception handler i.e. it is used to communicate information about error. A throw expression accepts one parameter and that parameter is passed to handler.

a. throw statement is used when we explicitly want an exception to occur, then we can use throw statement to throw or generate that exception.

Need of Exception Handling

- a simple example to understand the usage of try, catch and throw. Below program compiles successfully but the program fails at runtime, leading to an exception.

```
#include <iostream>
#include<conio.h>
int main()
{
    int a=10,b=0,c;
    c=a/b;
    return 0;
}
```

- The above program will not run, and will show **runtime error** on screen, because we are trying to divide a number with **0**, which is not possible.
- How to handle this situation? We can handle such situations using exception handling and can inform the user that you cannot divide a number by zero, by displaying a message.

Using try, catch and throw Statement

```
#include<iostream.h>
#include<conio.h>
void main()
{
 int n1,n2,result;
cout<<"\nEnter 1st number : ";
cin>>n1;
cout<<"\nEnter 2nd number : ";
cin>>n2;
```

```
try
    {
        if(n2==0)
            throw n2;          //Statement 1
        Else
         {
            result = n1 / n2;
            cout<<"\nThe result is : "<<result;
        }
    }

catch(int x)
    {
        cout<<"\nCan't divide by : "<<x;
    }
cout<<"\nEnd of program.";
}
```

Multiple catch blocks

- A **single try statement** can have multiple catch statements. Execution of particular catch block depends on the type of exception thrown by the throw keyword. If throw keyword send exception of integer type, catch block with integer parameter will get execute.

```
#include<iostream.h>
#include<conio.h>
void main()
{ int a=2;
      try
        {
            if(a==1)
               throw a;     //throwing integer exception
            else if(a==2)
               throw 'A';     //throwing character exception
            else if(a==3)
               throw 4.5;     //throwing float exception
         }
```

```
catch(int a)
    {
        cout<<"\nInteger exception caught.";
    }
catch(char ch)
    {
        cout<<"\nCharacter exception caught.";
     }
 catch(float d)
    {
        cout<<"\nFloat exception caught.";
     }
cout<<"\nEnd of program.";
}
```

Catch All Exceptions

- The above example will caught only three types of exceptions that are integer, character and float. If an exception occur of long type, double type, no catch block will get execute and abnormal program termination will occur. To avoid this, We can use the catch statement with three dots as parameter (...) so that it can handle all types of exceptions.

```
catch(...)
    {
       cout<<"\nException occur.";
     }
cout<<"\nEnd of program.";
```

References:

- www.studytonight.com
- www.tutorialpoint.com
- www.geeksforgeeks.org
- “Object oriented programming in C++” Robert Lafore
- “Object oriented programming with C++”, E.Balagurusamy