Object oriented programming using C++

BCA- 208

UNIT-4

Topic: Exception Handling

Jyoti Jain
Assistant professor
HIMT, Rohtak
E-mail: jyoti.goyal24@gmail.com

Meaning of Exception handling

- Exception is the place where a problem has occurred.
- Handling is the place for the solution to the specific problem.

Exception:

An exception is an event, which occurs during the execution of the program, that disrupts the normal flow of the program instructions.

Exception Handling in C++

- Exceptions are run-time anomalies or abnormal conditions that a program encounters during its execution.

There are two types of exceptions

- a)Synchronous
- b) Asynchronous(Ex: which are beyond the program’s control, Disc failure etc).

Handling the Exception

- Handling the exception is nothing but converting system error message into user friendly error message.

C++ provides following specialized keywords for this purpose.

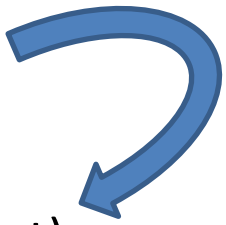
- *try*: represents a block of code that can throw an exception.
- *catch*: represents a block of code that is executed when a particular exception is thrown.
- *throw*: Used to throw an exception. Also used to list the exceptions that a function throws, but doesn't handle itself.

Syntax of exception handling

```
try
   {
        statements;
          ……..
    }
catch(type argument)
   {
        statements;
        ……..
   }
```

Syntax of exception handling

- Try
- {
- statements;
- …….
- Throw exception;
- }



- Catch(type argument)
- {
- statements;
- ……
- }

Why Exception Handling

- *Separation of Error Handling code from Normal Code*
- *Functions/Methods can handle any exceptions they choose*
- *Grouping of Error Types*

Simple program of Exception handling

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

int main()
{
int x = -1;

// Some code
cout << "Before try \n";
try {
        cout << "Inside try \n";
        if (x < 0)
        {
                throw x;
                cout << "After throw (Never executed) \n";
        }
}
catch (int x ) {
        cout << "Exception Caught \n";
}

cout << "After catch (Will be executed) \n";
return 0;
}
```

Exception and derived class

- To catch an exception for both base and derive class then we need to put catch block of derived class before the base class. Otherwise, the catch block of derived class will never be reached.

Here is a simple example where catch of derived class has been placed before the catch of base class, now check the Output

```
#include<iostream>
using namespace std;
class B {};
class D: public B {}; //class D inherit the class B
int main() {
  D derived;
  try {
    throw derived;
  }
  catch(D derived){
    cout<<"Caught Derived Exception"; //catch block of derived class
  }
  catch(B b) {
    cout<<"Caught Base Exception"; //catch block of base class
  }
  return 0;
}
```

Function Exception Declaration

- **C++** provides a mechanism to ensure that a given **function** is limited to throw only a specified list of **exceptions**.
- An **exception** specification at the beginning of any **function** acts as a guarantee to the **function's** caller that the **function** will throw only the **exceptions** contained in the **exception** specification.

Unexpected Exception

- When a function with an **exception** specification throws an **exception** that is not listed in its **exception** specification, the **C++** run time does the following: The **unexpected**() function is called. The **unexpected**() function calls the function pointed to by unexpected_handler .

Advantages of exception handling

- Exception handling helps the programmers to create reliable program.
- It separates the exception handling code from the main logic of the program.
- Grouping error types and error differentiation.

Thanks