Exception handling in C++ Design Patterns

Srivatsav Thumati ;
MS in CS; University of Bridgeport;
Bridgeport; USA

Tarek El Taeib;
University of Bridgeport
Bridgeport; USA

***Abstract*—In the world of C programming errors play a major role in the execution of a program. Whenever there is chance of an error there is a result of struggle to the programmer. Usually there are errors that might occur due to mistake in a code. But there are some errors that occur due to some arithmetical anomalies that occur due to some unanswered statements. The paper mainly discusses about this scenario where these type of errors called as runtime error occurs. This is called as exception handling. Types of exception handling and ways to bring the desired output is explained in the scenario of runtime errors is explained.**

Keywords—Error, C++, Exception, Try, Catch, Block

I. INTRODUCTION (*Heading 1*)

A C++ program is comprised of many types of error. Runtime errors occur while executing a program. Main purpose of the Exception handling mechanism is to provide means to detect and report an “exceptional circumstance” so that an appropriate action can be taken.

II. OVERVIEW ON EXCEPTIONAL HANDLING

A. *Types of errors in C++:*

In object oriented programming there are three types of errors. They are:

I. Syntax error

II. Logical error

III. Runtime error

Syntax error: This error occurs due to lack of language. Reasons for occurrence of this error are basic programming mistakes such as not providing optimistic number of opening and closing brackets, semicolon is not added at the termination of statement. Declaration of an identifier is missing, grammatical rule mistakes while declaration of an identifier.

IV. **Logical error:** This error usually occurs when there is a poor understanding of a problem and due to mistake in the solution. When this error occurs it will not be displayed on the screen. The results will be displayed wrong. One of the examples of this error is an infinite loop.

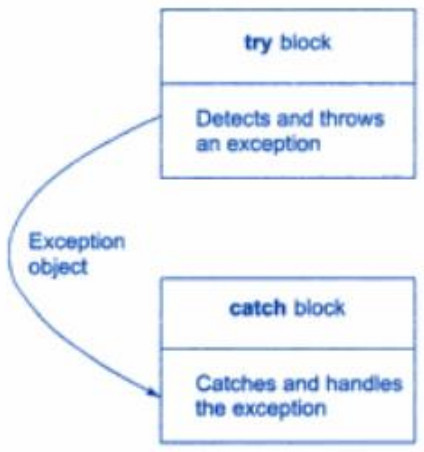
V. **Runtime exception:** This error occurs when output gets an undetermined value. The message would be overflow or underflow or Division by 0.Exceptions are some peculiar problems that may occur other than basic syntax and logical errors. For handling these exceptions there are built in language features provided by ANSI C++. Original C++ does not have the features of exception handling but this feature is present in ANSI C++.

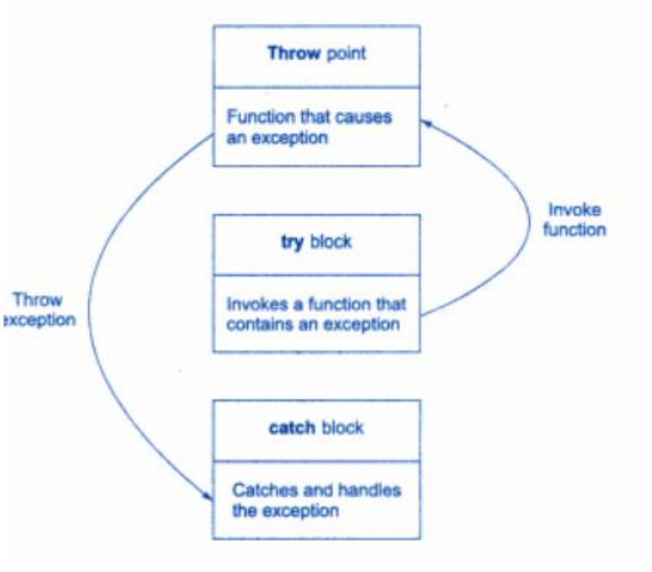
B. *Basics of Exception handling*

Exceptions are classified into two categories. They are synchronous exceptions and asynchronous exception. Errors that occur in synchronous exceptions are out of range index and over flow. Asynchronous exceptions are occurred when there is a phenomenon such as keyboard interrupts i.e., when the program cannot be controlled. An error handling code is provided separately in this mechanism. The tasks performed by error handling code provided are Finding and informing the problem, Receiving the information and taking corrective actions. There are two segments consisted in error handling code. One is to detect errors and the other is throw exceptions and to take appropriate actions against the errors.

III. EXCEPTION HANDLING MECHANISM

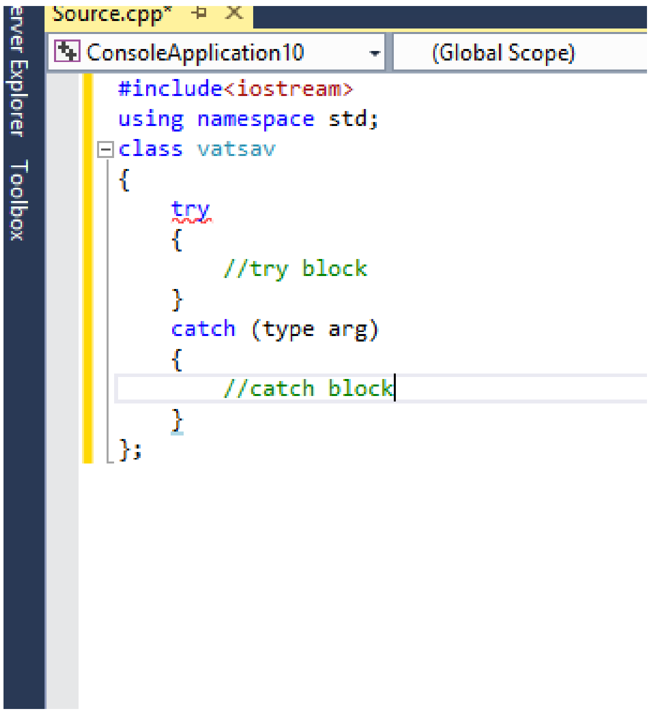
Basically there are three keywords in error handling code they are “try, throw and catch”. ‘Try’ keyword is used to preface a block of statements that generates exception. The statements are surrounded by braces. ‘catch’ catches and handles the exceptions that are thrown by ‘throw’ from try block as shown in the figure below. After the try block throws an exception the program enters the catch statement. Syntax of the exception mechanism is as shown in the figure below.


try block
Detects and throws an exception
Exception object
catch block
Catches and handles the exception


Throw point
Function that causes an exception
try block
Invokes a function that contains an exception
catch block
Catches and handles the exception
Invoke function
Throw exception

Finally, complete content and organizational editing before formatting. Please take note of the following items when proofreading spelling and grammar:

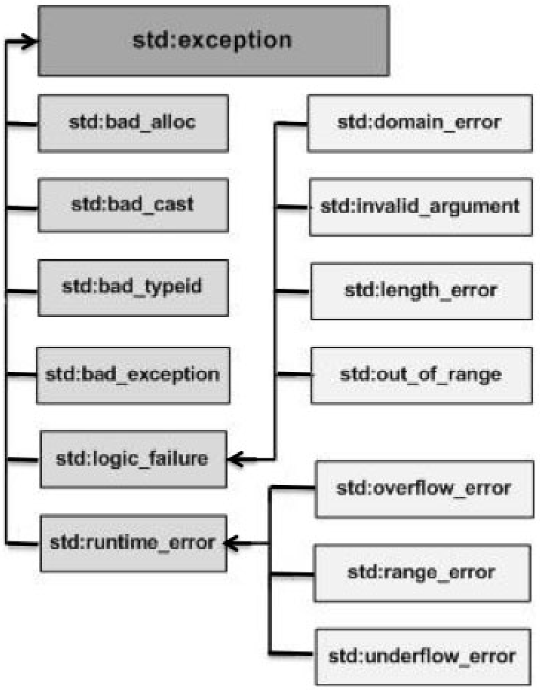
A. Example for TRY block exception



```
#include<iostream>
using namespace std;
class vatsav
{
    try
    {
        //try block
    }
    catch (type arg)
    {
        //catch block
    }
};
```

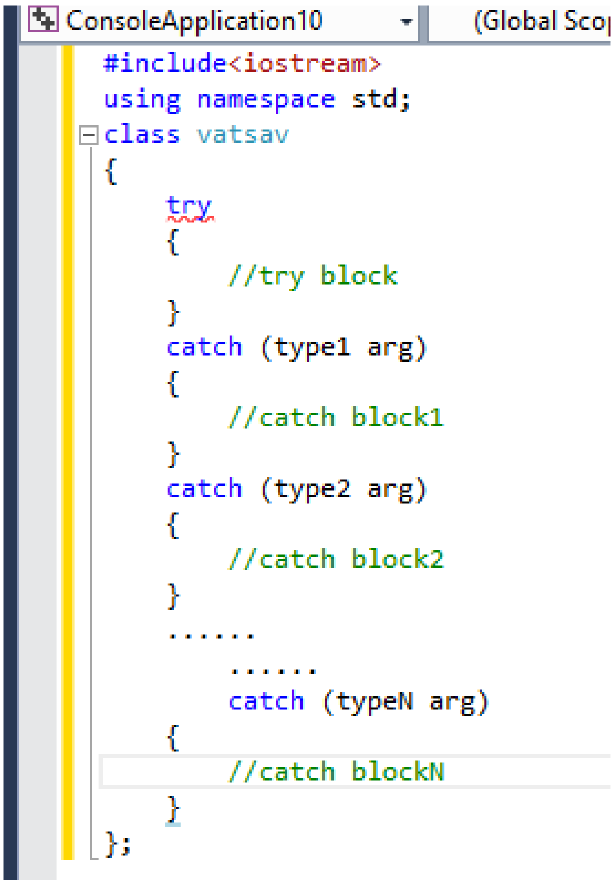
B. The throw exceptions thrown from c++ standard library to the exception class. These exceptions are as follows

Exception name is bad_alloc it is thrown by new on allocation failure, bad_cast its description is to thrown by dynamic_cast when it fails in a dynamic cast, bad_exception is thrown by certain dynamic exception specifiers, bad_typeid is thrown by type id, bad_function_call is thrown by empty function objects, bad_weak_ptr is thrown by shared_ptr when passed a bad weak_ptr.



- Two generic exception types are inherited by custom exceptions to report errors they are logic error and runtime error. Logic error is an error related to the internal logic of the program and runtime error is an error detected during runtime.

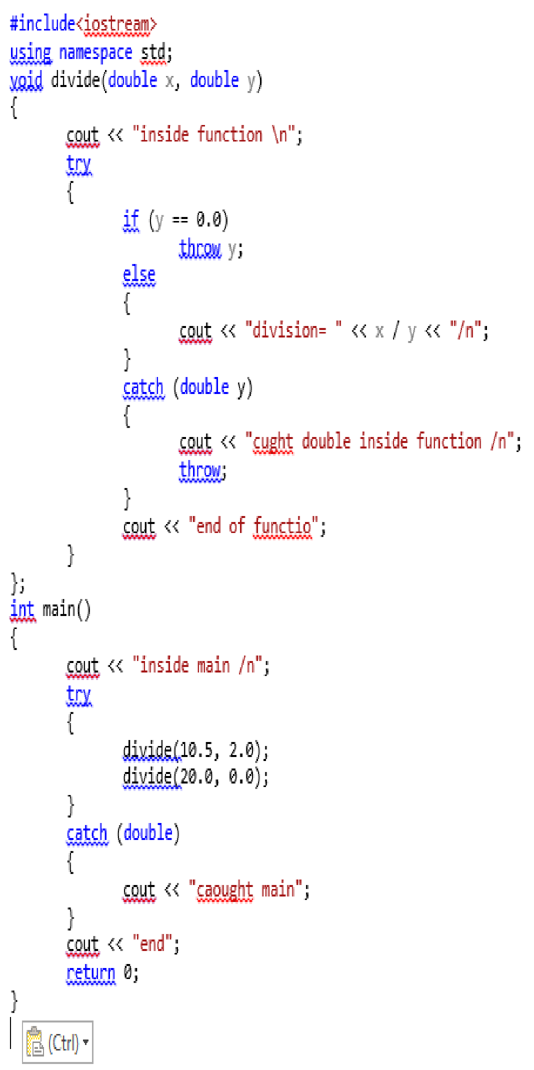
- Multiple catch statements: There is a chance of occurrence for a program segment to throw more than one condition to exception. Syntax of this condition is shown below.



```
#include<iostream>
using namespace std;
class vatsav
{
    try
    {
        //try block
    }
    catch (type1 arg)
    {
        //catch block1
    }
    catch (type2 arg)
    {
        //catch block2
    }
    ......
        ......
        catch (typeN arg)
    {
        //catch blockN
    }
};
```

C. *Rethrowing an Exception*

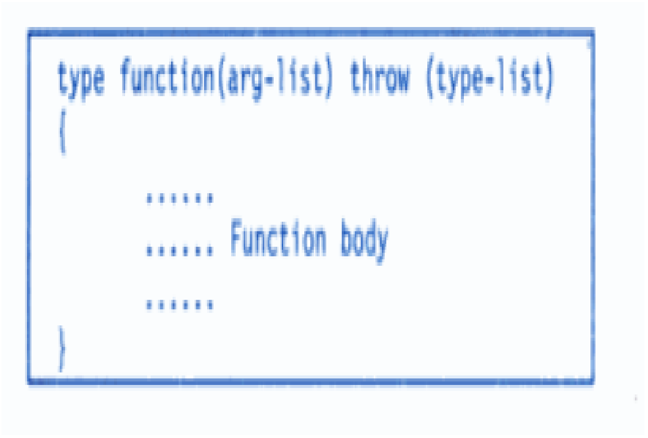
without processing an exception, a handler may decide to rethrow the exception that is caught. When an exception is rethrown it causes the exception to be thrown into the next try-catch sequence. This is explained below.



```
#include<iostream>
using namespace std;
void divide(double x, double y)
{
        cout << "inside function \n";
        try
        {
                if (y == 0.0)
                        throw y;
                else
                {
                        cout << "division= " << x / y << "/n";
                }
                catch (double y)
                {
                        cout << "cught double inside function /n";
                        throw;
                }
                cout << "end of functio";
        }
};
int main()
{
        cout << "inside main /n";
        try
        {
                divide(10.5, 2.0);
                divide(20.0, 0.0);
        }
        catch (double)
        {
                cout << "caought main";
        }
        cout << "end";
        return 0;
}
```

D. *Specifying Exception*

If needed a function can throw specified exceptions. By adding a throw list clause to the function definition it is achieved and its syntax is given in the below figure.



```
type function(arg-list) throw (type-list)
{
        ......
        ...... Function body
        ......
}
```

IV. CONCLUSION

Exception handling Is one of the advantages of c++ over c. Some of the major advantages of exception handling is the separation of error handling code from normal code. Functions can handle any exception they choose and grouping of error types.

REFERENCES

Object oriented programming with c++ 5th edition by balaguruswamy

1. C++ Exception handling by Christophe de Dinechin

2. http://cpluspluslearner.blogspot.com/2012/05/types-of-errors-in-c_11.html

3. http://www.geeksforgeeks.org/exception-handling-c/

4. http://www.cplusplus.com/doc/tutorial/exceptions/

5. http://www.tutorialspoint.com/cplusplus/cpp_exceptions_handling.htm