# EXCEPTION HANDLING

Pranali P Chaudhari

# Contents

- Errors and Exception
- Exception Handling Mechanism
  - Try, Throw and Catch
- Re-throwing an Exception
- Specifying Exceptions

# Quiz 1

- What is an error?
  - An error is a term used to describe any issue that arises unexpectedly and results in incorrect output.
- What are the different types of errors?
  - Logical error:
    - Occur due to poor understanding of problem or solution procedure.
  - Syntactic error:
    - Arise due to poor understanding of the language itself.
- What is an exception?
  - Exceptions are run time anomalies or unusual conditions that a program may encounter while executing.

# Exception Handling

- Exceptions are of two types:
  - **Synchronous exceptions**
    - The exceptions which occur during the program execution due to some fault in the input data are known as synchronous exceptions.
    - For example: errors such as out of range, overflow, underflow.
  - **Asynchronous exceptions.**
    - The exceptions caused by events or faults unrelated (external) to the program and beyond the control of the program are called asynchronous exceptions.
    - For example: errors such as keyboard interrupts, hardware malfunctions, disk failure.

# Exception Handling Mechanism

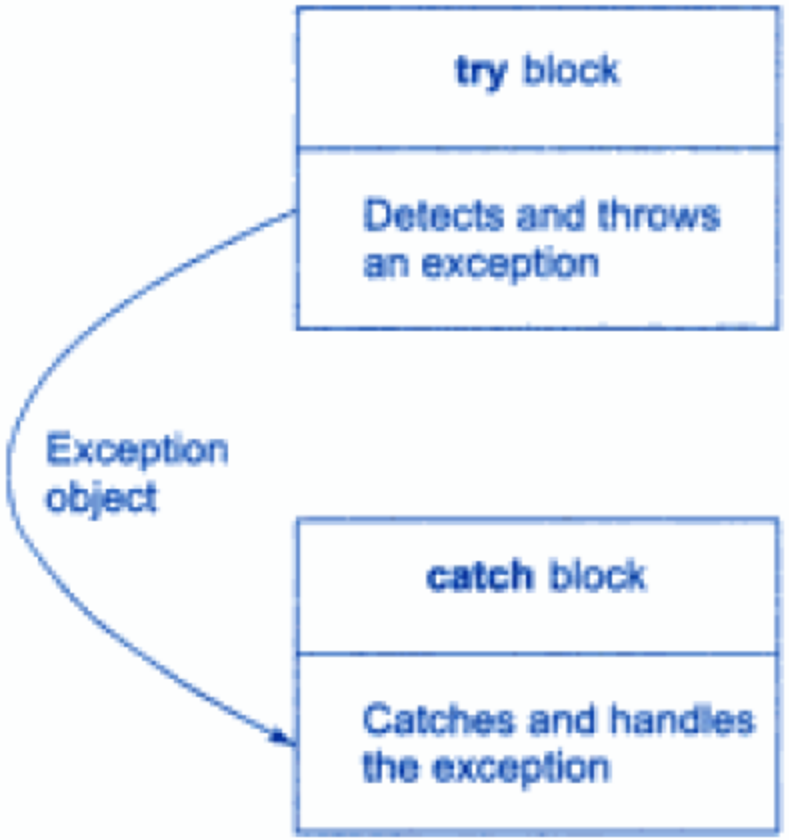
- Exception handling mechanism provides a means to detect and report an exception circumstances.
  - Find the problem (Hit the exception)
  - Inform that an error has occurred (Throw the exception)
  - Receive the error information (Catch the exception)
  - Take corrective actions (Handle the exception)
- The error handling consists of two segments

# Exception Handling Mechanism

- The exception handling mechanism is built upon three keywords:
  - Try
    - Is used to preface a block of statements which may generate exceptions.
  - Throw
    - When an exception is detected, it is thrown using a throw statement in the try block.
  - Catch
    - A catch block defined by the keyword catch catches the exception thrown by the throw statement in the try block and handles it appropriately.

# Exception Handling Mechanism

- When the try block throws an exception the program control leaves the try block and enters the catch statement of the catch block.
- If the type of object thrown matches the arg type in the catch statment the catch block is executed.
- Otherwise the program is terminated with the help of abort( ) function.

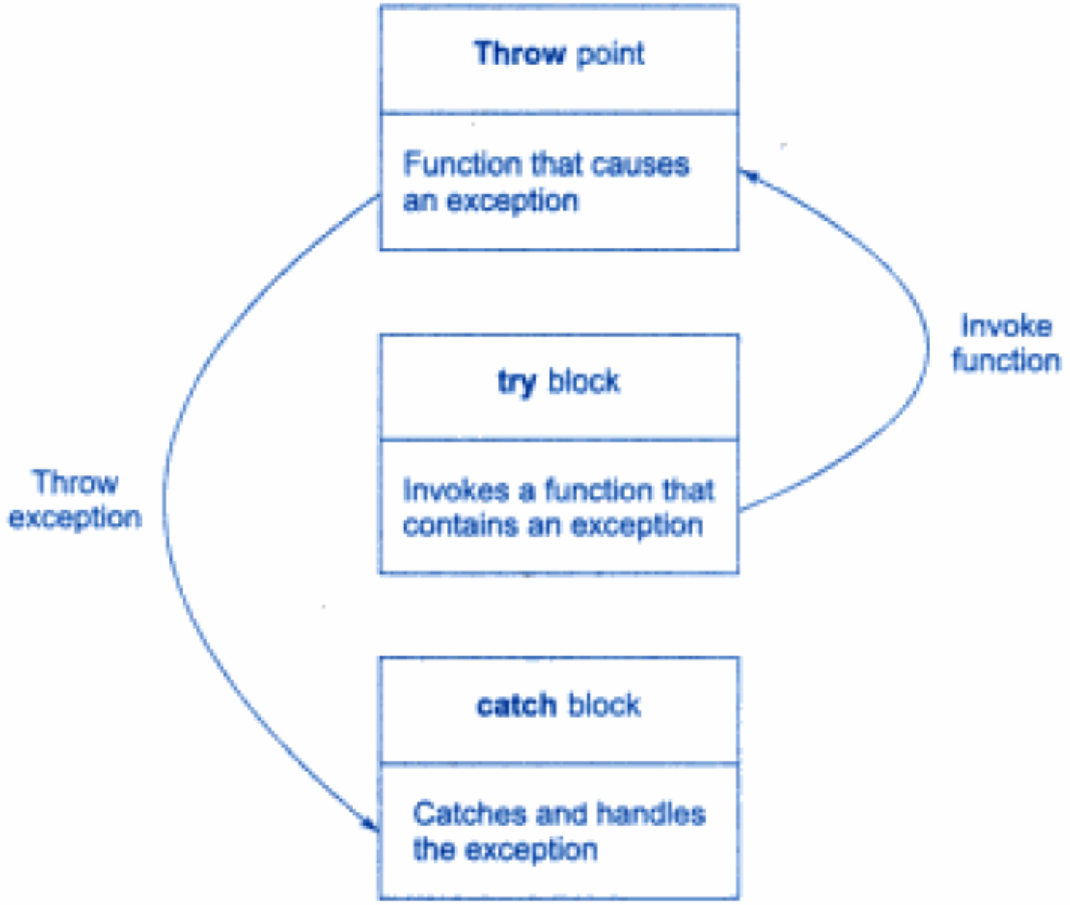


# Try block throwing an exception

```
int main()
{
    int a,b;
    cout<<“enter the values of a
    and b :”;
    cin>>a;
    cin>>b;
    int x = a-b;
    try
    {
        if(x != 0)
        {
          cout<<“Result (a/x) =“
              << a/x;
        }
        else
        {
            throw(x);
        }
    }
    catch(int i)
    {
      cout<<“Exception Caught
    :           x = “ << x << “\n”;
    }
    return 0;
}
```

# Exceptions thrown by functions

- Mostly exceptions are thrown by functions that are invoked from within the try blocks.
- The point at which the throw is executed is called the throw point.



# Exceptions thrown by functions

```
void divide(int x, int y, int z)
{
   if((x-y) != 0)
   {
   int R = z/(x-y);
   cout << “Result = “ << R << “\n”;
   }
   else
   {
   throw (x-y);
   }
}
```

# Exceptions thrown by functions

```
int main()
{
    try
    {
        divide(10,20,30);
        divide(10,10,20);
    }
    catch(int i)
    {
        cout << “\n Exception caught” ;
    }
    return 0;
}
```

# Throwing Mechanism

- When an exception is desired to be handled is detected, it is thrown using the throw statement.
- Throw statement has one of the following forms:
  - throw(exception);
  - throw exception;
  - throw;
- The operand object exception may be of any type, including constants.

# Catching Mechanism

- A catch block looks like a function definition:

```
catch(type arg)
{
    // statements for managing exceptions.
}
```

- The type indicates the type of exception that catch block handles.
- The catch statement catches an exception whose type matches with the type of catch argument.

# Multiple Catch Statements

- Multiple catch statements can be associated with a try block.
- When an exception is thrown, the exception handlers are searched for an appropriate match.
- The first handler that yields the match is executed.
- After executing the handler, the controls goes to the first statement after the last catch block for that try.

# Multiple Catch Statements

```
void test(int x)
{
   try
   {
   if (x==1) throw x;
   else
   if(x==0) throw ‘x’;
   else
   if(x== -1) throw 1.0;
   cout<<“\nEnd of try-block”;
   }
}
```

```
catch(char c)     // catch 1
{
   cout<<“\nCaught a character”;
}
catch(int m)  // catch 2
{
   cout<<“\nCaught an integer”;
}
catch(double d)        // catch 3
{
   cout<<“\nCaught a double”;
}

cout<<“\n End of try-catch block”;
```

# Multiple Catch Statements

```
int main( )
{
   cout<<“\n x = =1”;
   test(1);
   cout<<“\n x = = 0”;
   test(0);
   cout<<“\n x = = -1”;
   test(-1);
   cout<<“\n x = =2”;
   test(2);
   return 0;
}
```

```
x == 1
Caught an integer
End of try-catch system

x == 0
Caught a character
End of try-catch system

x == -1
Caught a double
End of try-catch system

x == 2
End of try-block
End of try-catch system
```

# Catch all Exceptions

- Sometimes it is not possible to anticipate all possible types of exceptions and therefore not able to design independent catch handlers to catch them.
- A catch statement can also force to catch all exceptions instead of a certain type alone.
- Syntax:

```
catch (…)
{
// statements for processing all exceptions.
}
```

# Catch all Exceptions

```
void test(int x)
{
    try
    {
        if (x==1) throw x;
        else
        if(x==0) throw 'x';
        else
        if(x== -1) throw 1.0;
        cout<<"\nEnd of try-
    block";
    }
}
catch(…)
{
    cout<<"\n Caught an
    exception";
}
```

```
int main( )
{
    cout<<"\nTesting generic
    catch";
    test(1);
    test(0);
    test(-1);
    test(2);
    return 0;
}
```

# Re-throwing an Exception

- A handler can re-throw the exception caught without processing it.
- This can be done using throw without any arguments.
- Here the current exception is thrown to the next enclosing try/catch block.
- Every time when an exception is re-thrown it will not be caught by the same catch statements rather it will be caught by the catch statements outside the try catch block.

# Re-throwing an Exception

```
void divide(double x, double y)
{
 cout<<“Inside Function”;
 try
 {
      if(y = =0.0)
 throw y;
      else
 cout<<“Division = “ <<x/y<<“\n”;
 }
  catch(double)
 {
       cout<<“\nCaught double inside function”;
       throw;
 }
 cout<<“\n End of function”;
}
```

```
int main()
{
 cout<<“\n Inside main”;
 try
 {
 divide(10.5, 2.0);
 divide(20.0, 0.0);
 }
 catch(double)
 {
 cout<<“\n Caught double    inside
 main”;
 }
 cout<<“\n End of main”;
 return 0;
}
```

# Specifying Exceptions

- It is possible to restrict a function to throw only certain specified exceptions.
- This is done by adding a throw list clause to the function definition.

  type function(arg-list) throw (type-list)

  {

  .....

  .....

  }
- The type-list specifies the type of exceptions that may be thrown.
- Throwing other type of exceptions cause abnormal termination of program.

# Specifying Exceptions

```
void test(int x) throw (int, double)
{
 if (x==0) throw 'x';
 else
 if(x==1) throw x;
 else
 if(x== -1) throw 1.0;
 cout<<"\n End of function block";
}
int main( )
{
 try
 {
 cout<<"\nTesting  throw restrictions";
 cout<<"\n x==0";
 test(0);
 cout<<"\n x==1";
 test(1);
 cout<<"\n x== -1";
 test(-1);
 cout<<"\n x== 2";
 test(2);
 }
```

```
Catch(char c)
{
    cout<<"\n Caught a character";
}

Catch(int m)
{
    cout<<"\n Caught a integer";
}

Catch(double d)
{
    cout<<"\n Caught a double";
}

Cout<<"\n End of try catch block";

return 0;
```

}

# Summary

- _____ are peculiar problems that a program may encounter at run time.
- Exceptions are of two types _____ and _____.
- An exception is caused by a faulty statement in ____ block, which is caught by _____ block.
- We can place two or more catch blocks to catch and handle multiple types of exceptions. (True/ False).
- It is also possible to make a catch statement to catch all types of exception. (True/ False)
- We cannot restrict a function to throw a specified exceptions. (True /

# Short Answer Questions

- What is an exception?
  - Exceptions are run time anomalies or unusual conditions that a program may encounter while executing.
- How is exception handled in C++?
  - In C++ the exception is handled using the three keywords try, throw and catch. Or try-catch mechanism.
- What are the advantages of using exception handling mechanism in a program?
  - The purpose of exception handling mechanism is to provide a means to detect and report an exceptional circumstances so that appropriate action can be taken and prevent abnormal termination of program.

# Short Answer Questions

- When should a program throw an exception?
  - There are some situation when a program come across unexpected errors and cause abnormal termination of program. To handle such errors and prevent program from termination exceptions are thrown and handled.
- What should be placed inside the try block?
  - The statement that may generate an exception are placed in the try block.
- When do we use multiple catch handlers?
  - Multiple catch handlers are used in a situation where a program has more than one condition to throw and exception.

# Short Answer Questions

- Explain under what circumstances the following statements would be used:
  - throw;
    - Re-throwing an exception.
  - void fun1(float x) throw()
    - Prevent a function from throwing any exception.
  - catch( ... )
    - Used to catch all types of exceptions.

# References

- Object Oriented Programming with C++ by E. Balagurusamy.

# END OF UNIT ....