Exception Handling

Types of Errors…

- Whenever we write a program we may encounter two kinds of errors such as:

  1.*Syntax Error:* It is due to poor understanding of the language.
  2.*Logical Error:* It is due to poor understanding of the problem.

  Apart from these two kinds of error, we have some kind of a problem called ***exception***.

Exception Handling Mechanism

- Exception Handling mechanism is used to provide the way to detect and report an "*exceptional circumstance*" so that appropriate action can be taken.
- C++ exception is basically based on three keywords:
    - try
    - throw
    - catch

Exception Handling Mechanism(cont….)

try

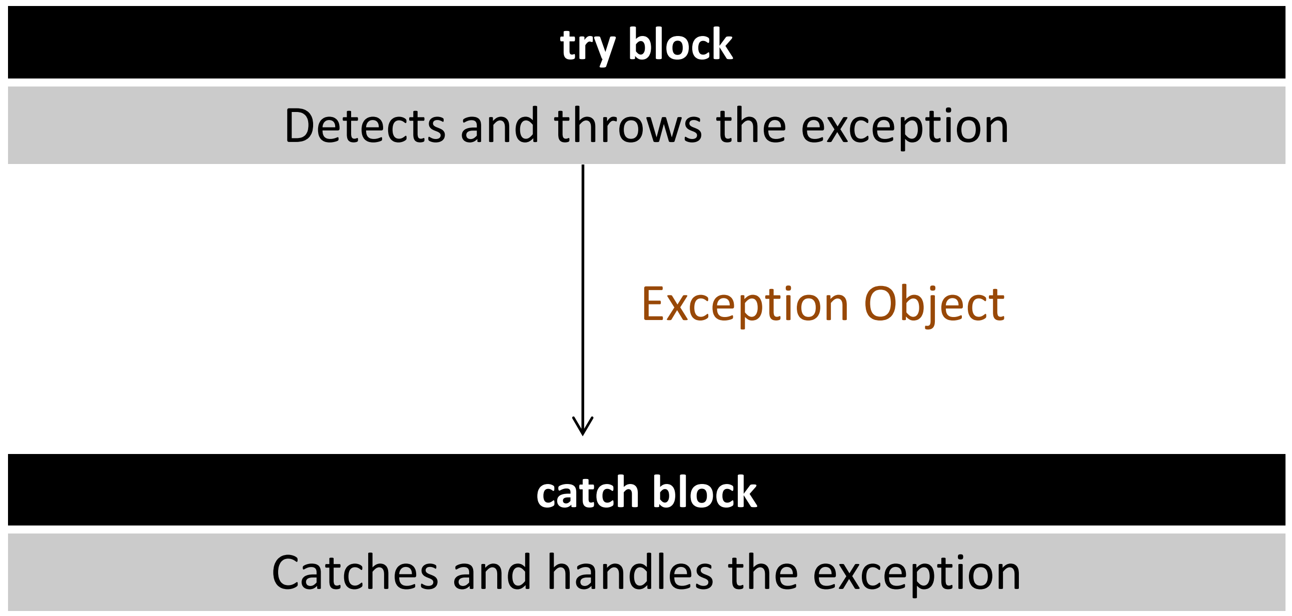
This keyword is used to write a block of statement which may generate exceptions. This block is called ***try block***.

throw

If an exception is found it is thrown using ***throw statement***.

catch

This keyword catches the expression thrown by the throw statement from the try block and handle it properly. This block is called ***catch block.***


try block
Detects and throws the exception
Exception Object
catch block
Catches and handles the exception

Syntax:

```
try
{
        ....
        throw exception;
        ....
}
catch(type argument)
{....}
```

Things to Remember…

- When try block throws an exception the program control leaves the try block and enters the catch statement of the catch block of the type of exception matches with the argument type of catch , then catch block is executed for handling the exception.
- If they don’t match the program is terminated with the help of *abort()* function which is invoked implicitly by default.
- When no exception is detected, the control goes to the statement immediately after the catch block i.e. the catch block is skipped.

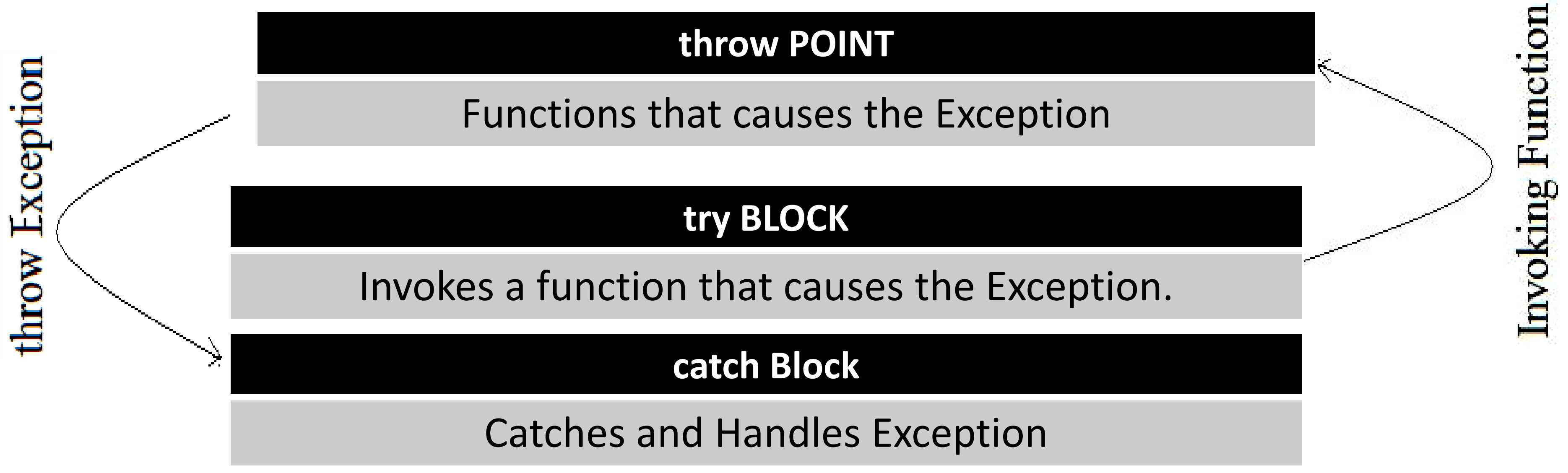
Example

```
void main()
{
    int a, b;
    cout<<"Enter the 2 values for
    a, b:: ";
    cin>>a>>b;
    try
    {
        if(b!=0)
        {
            cout<<a/b;
        }
    else
        throw b;
    }

    catch(int)
    {
        cout<<"Exception Caught.";
    }
}
```

Function Generating a try Block or Function Generating an Exception

Most often exceptions are thrown by function that are invoked from within the try block called *throw point.* Once an exception is thrown to the catch block control cannot return the throw point.


throw POINT
Functions that causes the Exception
try BLOCK
Invokes a function that causes the Exception.
catch Block
Catches and Handles Exception
throw Exception
Invoking Function

Syntax

```
return type<function name><arg>
{
    ….
    ….
    throw exception
    ….
    ….
}
```

```
try
{
    …
    invoking function;
    …
}
catch(type arg)
{
    Handle the Exception;
}
```

Note: The catch block immediately follows the try block irrespective of the location of the throw point.

Example

```
#include<iostream.h>
void divide(int a, int b)
{

    if(b!=0)
    {
          cout<<a/b;
    }

    else
          throw b;

}
```

```
int main()
{
          try{
          cout<<"We are in try
block..."<<endl;
                    //divide(10,0);
                    divide(15,5);
                    divide(10,0);

          }
          catch(int)
          {
          cout<<endl<<"Exception Caught...";
          }
return 0;
}
```

```
O/P:  We are in try block...
       3
      Exception Caught
```

Multiple Catch

Syntax

```
try
{
    throw(exception object)
}
catch(type arg)
{…}
catch(type arg)
{…}
catch(type arg)
{…}
……
……
```

Example

```
#include<iostream.h>

void test(int x)
{
    try
    {
    if(x==0)
            throw 'x';

    else if(x==1)
    {
            throw 1.0;
    }
    else if(x==2)
            throw x;
    }
    }
```

```
catch(int)
    {
            cout<<"INT"<<endl;
    }

    catch(char)
    {
    cout<<"CHAR"<<endl;
    }

    catch(double)
    {
    cout<<"DOUBLE"<<endl;
    }
}
```

Cont…

```
int main()
{
    test(0);
    test(1);
    test(2);
return 0;
}
```

O/P:

```
CHAR
DOUBLE
INT
```

In the above program , more than one catch block is there and based on the match the appropriate block will be executed.

Generic Catch

In some situations, we may not be able to anticipate all possible types of exceptions and therefore may not be able to catch them. In that case, it is possible to force a catch statement to catch all exceptions irrespective of the type of exception.

Syntax

```
catch(…)
{
        …..
        …..
}
```

Example

```
O/P
        CAUGHT ALL
        CAUGHT ALL
        CAUGHT ALL
```

```
void test(int x)
{
    try
    {
    if(x==0)
            throw 'x';

    else if(x==1)
    {
            throw 1.0;
    }
    else if(x==2)
            throw x;
    }
```

```
 catch(...)
        {
        cout<<"CAUGHT
ALL"<<endl;
        }

}

int main()
{
        test(1);
        test(2);
        test(0);
         return 0;
}
```