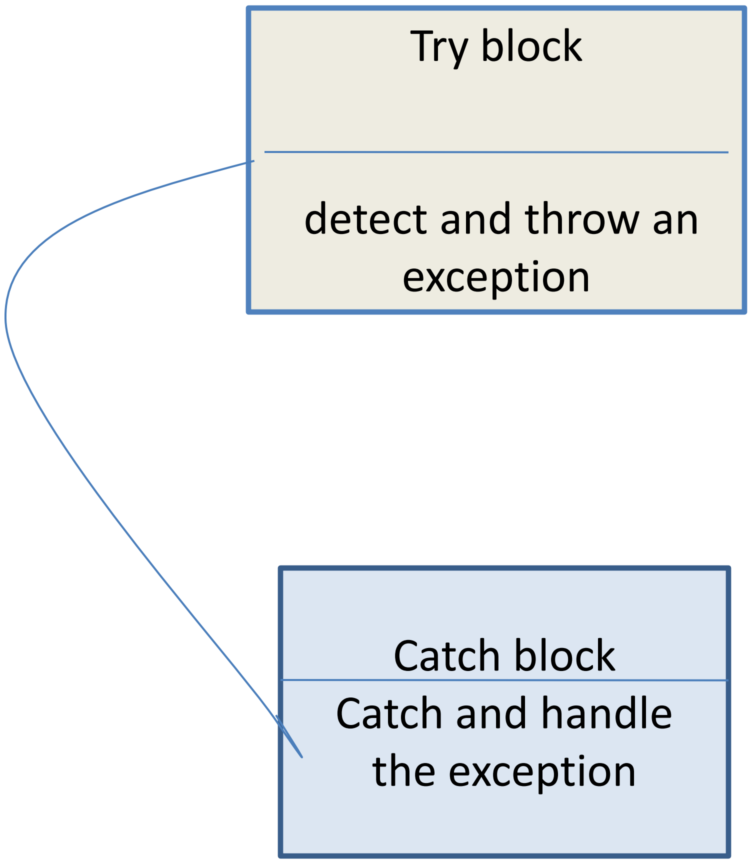
UNIT-4-C

Introduction, Basics of C++ Exception Handling: Try Throw, Catch, Throwing an Exception, Catching an Exception, Rethrowing an Exception, Exception specifications, Processing Unexpected Exceptions, Stack Unwinding, Constructors, Destructors and Exception Handling, Exceptions and Inheritance.

Exception Handling

- Exceptions are the run time anomalies or unusual conditions that a program may encounter while executing.
- Exception handling is not a part of original C++.
- It provides a type safe & an integrated approach for coping with unusual predictable problems that arise while executing a program.
- This mechanism is based on 3 keywords namely:
- try
- Throw
- Catch
- Try :- is used to preface a block of statements (surrounded by braces) which may generate exceptions. Its called a try block.
- Throw :- when the exception is detected ,it is thrown using a throw statement in try block.
- Catch :- this keyword ‘catches’ the exception ‘thrown’ by the throw statement in the try block and handles the exception appropriately.


Try block
detect and throw an exception
Catch block
Catch and handle the exception

Exception and Exception Handlers

Exception Handling –

It is a mechanism to detect and report an ‘exceptional circumstance” so that appropriate action can be taken. It involves the following tasks.

- **Find the problem (Hit the exception)**
- **Inform that an error has occurred (Throw the exception)**
- **Receive the error information (catch the expression)**
- **Take corrective action (Handle the exception)**

```
main()
{  int x, y;
    cout << “Enter values of x and y”;
    cin >>x>>y;
   try {
            if (x != 0)
                cout “y/x is =“<<y/x;
            else
                 throw(x);
      }
  catch (int i)  {
     cout << “Divide by zero exception caught”;
    }
}
```

Exception and Exception Handlers

try – Block contains sequence of statements which may generate exception.

throw – When an exception is detected, it is thrown using throw statement

catch – It's a block that catches the exception thrown by throw statement and handles it appropriately.

catch block immediately follows the try block.

The same exception may be thrown multiple times in the try block.

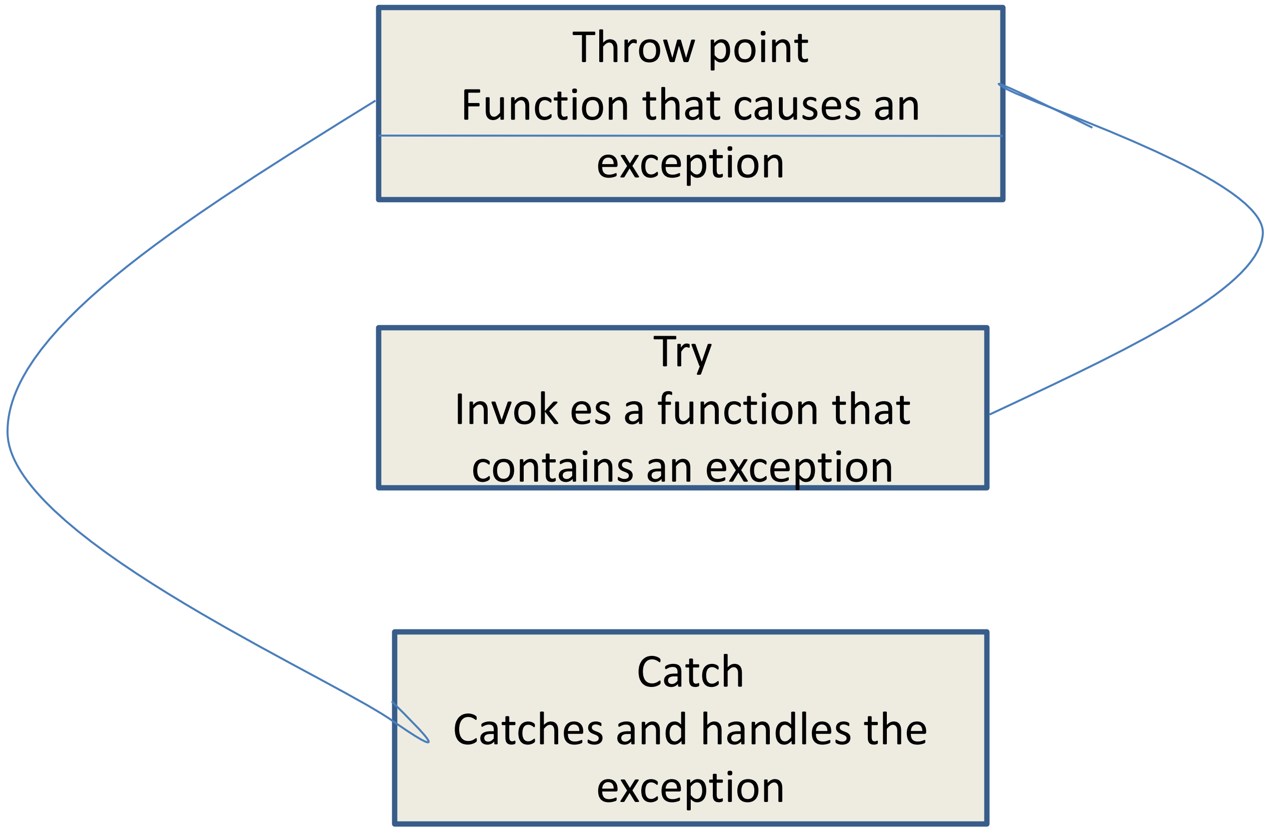
There may be many different exceptions thrown from the same try block.

There can be multiple catch blocks following the same try block handling different exceptions thrown.

The same block can handle all possible types of exceptions.

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

Function invoked by try block throwing exception


Throw point
Function that causes an exception
Try
Invok es a function that contains an exception
Catch
Catches and handles the exception

Invoking function that generates exception

```
Void divide(int x,int y)
{
if (x != 0)
                cout “y/x is =“<<y/x;
            else
                 throw(x);
}

Void main()
{  int x, y;
    cout << “Enter values of x and y”;
    cin >>x>>y;
   try {
       cout<<“try block”
      divide(10,20);
      divide(0,20);
                }
  catch (int i)  {
     cout << “Divide by zero exception caught”;
    }
}
```

Throwing machanism

Throw statement is in one of the following form:

- Throw(exception)
- Throw exception
- Throw   //rethrowing an exception

Catching machanism

```
Catch(type arg)
{
   //statement for managing exceptions
}
```

Multiple catch statement(one try block)

```
  void test(int x)
{
    try
   {
    if(x==1) throw x;
     else if(x==0)  throw ‘x’;
     else if (x==-1) throw  1.0;
     else  cout<<“end of try block”;
    }
  Catch(char c){cout<<“char”; }
  Catch(int m){cout<<“int”; }
  Catch(double d){cout<<“double”;  }
  cout<<“end of function”;
}
int main()
{
Cout<“testing multiple catch”;
Cout<<“x==1”;
Test(1);
Cout<<“x==0”;
Test(0);
Cout<<“x==-1”;
Test(-1);
Cout<<“x==2”;
Test(2);}
```

```
Output:-
Testing multiole catch
X==1
Int
X==0
Char
X==-1
Double
End of try block
End of function
```

Catching all exceptions

```
void test(int x)
{
    try
   {
    if(x==1) throw x;
    if(x==0)  throw ‘x’;
    if (x==-1) throw  1.0;
  }
   Catch(…)
{
cout<<“caught an exception”;
}
 void Main()
{
Cout<“testing”;
Test(-1);
Test(0);
Test(1);
}
```

Output:-

```
Testing
Caught an exception
Caught an exception
Caught an exception
```

Rethrowing an exception(throw)

```
Void divide(int x,int y)
{
            try{
            if (x != 0)
                        cout "y/x is ="<<y/x;
                    else
                          throw(x);
            }
            Catch(int )
            {  cout<<"catch int inside function";
               throw;
            }
}
Void main()
{
      try
      {
                cout<<"try block"
                divide(10,20);
                divide(0,20)
            }
             catch (int )  {
             cout << "catch int inside main ";
               }
}
```

```
Output:-
Try block
y/x is=2
catch int inside function
catch int inside main
```

Specifying exception

- It is possible to throw only certain specified exceptions. Syntax is:

```
Type   function(arg-list)  throw (type-list)
{
………..                    //function body
……………
}
```

If we wish to prevent a function from throwing any exception ,we must use:

```
throw();
```

```
void test(int  x)  throw(int,double)
{
     if(x==1) throw x;
     else if(x==0)  throw ‘x’;
     else if (x==-1) throw  1.0;
     else  cout<<“end of function block”;
 }
```

```
Void  main()
T          try
            {
         Cout<“testing multiple catch”;
        Cout<<“x==0”;
        Test(0);
        Cout<<“x==1”;
        Test(1);
        Cout<<“x==-1”;
        Test(-1);
        Cout<<“x==2”;
        Test(2);
        }
         Catch(char c){cout<<“char”;}
         Catch(int m){cout<<“int”; }
         Catch(double
        d){cout<<“double”;}
          Cout<<“end of try catch
        system”
}
```

```
Output:-testing multiple catch
                end of try catch system
```

Note: Here throwing any other type of exception will cause abnormal program termination