

Exception handling

- An exception is an unwanted or unexpected event, which occurs during the execution of a program which disrupts the normal flow of the program's instructions.
- While execution of program an abnormal condition may occur, then following things possible
 - When an abnormal condition is generating in running process, execution of program get stopped and giving **run time error** message.

- What is the difference between error(s) and exception (s)?

- If abnormal condition does not handle

▪ Process -> **Abnormal Condition** -> **Run Time Error**

- Once a Run Time Error Occurred program will be terminate forcefully.

- If abnormal condition handle

▪ Process -> **Abnormal Condition** -> **Exceptionally Handled**

Process continue

- What is exactly happens when an abnormal condition is generated?

- When an abnormal condition is generate in process, C++ technology throws an **exception object** to the process and if the programmer does not handle the abnormal situation then **exception object** makes "U-Turn" to c++ technology and goes to the "**default Handler**" of the c++ technology then **program suddenly terminate**.

- Some of more common ones are

- Falling short of memory
- Inability of opening file
- Exceeding the array out of bound
- Attempting to initialize an object to an impossible value
- Divided by 0
- Fatal error etc...

- Exceptions are Bugs that are handled at run time.

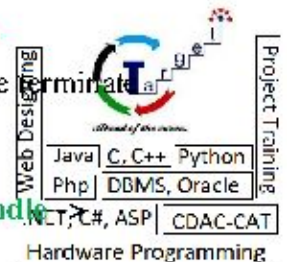
- C++ provides a systematic, object-oriented approach to handling runtime error generated by using classes.

- The exception mechanism of c++ uses three keywords

- try
- catch
- throw

- e.g.

```
#include<iostream>
using namespace std;
class Demo
{
public:
static Float divide (int a , int b )
{
return a/b ;
}
```



```

    }
};
main ()
{
    int x , y ;
    float z ;
    cout << "\nEnter a value of a and b ";
    cin >> x >> y;
    z = Demo::divide(x,y);
    cout << "\ndivide=" << z;
    cout << "\ncontinue program";
}

```

Enter a value of a and b 5

0

Floating point error: Divide by 0.

Abnormal program termination

- we can handle above abnormal condition

```

e.g.
#include<iostream.h>
class Demo
{
    public:
    static float divide (int a , int b )
    {
        return (float)a/b ;
    }
};
main ()
{
    int x , y ;
    float z ;
    cout << "\nEnter a value of a and b ";
    cin >> x >> y;
    try
    {
        z = Demo::divide(x,y);
        cout << "\ndivide=" << z;
    }
    catch(...)
    {
        cout << "undefined";
    }
    cout << "\ncontinue program";
}

```

- use exception handler for handling the exceptions

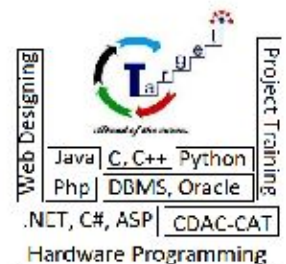
- try...catch

- syntax:

```

try
{
    .....
    .....
}

```



CDAC - CAT
webDesigning
Java. MS.net
Oracle, Training
Python, C, C++
Akhillesh Gupta
9981315087

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

e.g.

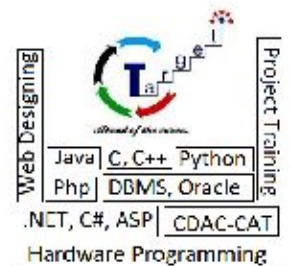
```
try
{
    z = Demo::divide(x,y);
    cout << "divide=" << z;
}
catch(...)
{
    cout << "undefined";
}
```


- **try and catch used in a pair.**
- **We can use any number of cache blocks with try block**

```
try
{
    ...
}
catch(Exception e )
{
    ...
}
catch(IOException e)
{
    ...
}
catch(...)
{
    ...
}
```

- **throwing an exception object**

```
#include<iostream>
using namespace std;
class Exp
{
};
class Demo
{
public:
    static float divide (int a , int b )
    {
        if ( b != 0 )
            return (float)a/b ;
        else
            throw Exp();
    }
};
```



CDAC - CAT
 **webDesigning**
Java, MS.net
Oracle, Training
Python, C, C++
Akhillesh Gupta
9981315087

```
main ()
{
    int x , y ;
    float z ;
    cout << "\nEnter a value of a and b ";
    cin >> x >> y;
    try
    {
        z = Demo::divide(x,y);
        cout << "\ndivide =" << z;
    }
    catch(Exp e)
    {
        cout << "undefined";
    }
    cout << "\ncontinue program";
}
```

Tips

- It is not necessary that the statement that causes an exception be located directly in the try block. It may as well be present in a function that is being called from try block.
- We can use any number of exception handler.
- We can use more than one exception handler for one try.


```
try
{
}
catch(Exp1)
{
}
catch(Exp2)
{
}
```

- If an exception other than the once specified in the exception specification is thrown then a **special function called unexpected() get called**.
- you can write your own version of unexpected() function by setting it up using **set_unexpected()** function.

e.g.

```
set_unexpected(my_unexpected);
```

- try block can be nested**
- if an exception is throws before constructor's execution is completed the associated destructor will not be called for that object
- When an exception is thrown , the exception -handling system looks through the nearest handler (Catch block)in order in which they are written. When it find a match the exception in considered handled and no further searching takes place.
- Do not place those statements that have errors in the handler.
- Do not overuse it.**



ahead of the curve.

Web Designing	Java	C, C++	Python	Project Training
	Php	DBMS, Oracle		
	.NET, C#, ASP			
	CDAC-CAT			

Hardware Programming



ahead of the curve.

CDAC - CAT

webDesigning

Java, MS.net

Oracle, Training

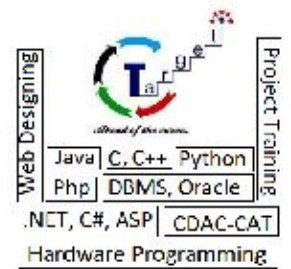
Python, C, C++

Akhillesh Gupta

9981315087



The Target Institute



CDAC - CAT
webDesigning
Java, MS.net
Oracle, Training
Python, C, C++
Akhillesh Gupta
9981315087