Exception Handling & Functions (unit-5)

2 ways for ex handling in a fun.

1) fun call can be placed inside a try catch block.

2) try catch block can be placed inside a fun.

1) Fun inside a try catch block.

```
int main()
{
    try          // fun calls r placed i/s the
                    try block.
    {
        fun(10); // No exception
        fun(0); // exception thrown. jump to
                        catch block.
        fun(100); // This part (third fun call) will
                        never run.
    }
    catch (int x)
    {
        cout<<"err occured";
        cout<<"err code"<< x;

    }

}
void fun(int x)
{
    if(x==0) throw 100;   } fun def.ⁿ It
                            will throw an
                            ex when value
                            is 0.
}
```

explanation :-

- In this case, ex will occur in second fun call, so the third fun call is never run.
- So fun call shud not be placed inside try block if mul fun call r made. bcoz the part after II fun ~~ca~~ call will not run.
- Instead the try & catch shud be placed i/s the fun ~~the~~ itself as shown in next eg.

---

2) try & catch block inside a fun

```
void fun(int x)
{
      try
      {
            if(x==0) throw 100;

      }
      catch(int x)
      {
            ⋮
      }
}
```

```
int main()
{
        fun(10); //NO ex
        fun(0);  // ex will be there but
                       it will be handled
                              i/s the fun
        fun(100);
                 // this fun call
                   will also run in this
                   case.

}
```

- In this eg., <u>all the fun call will</u> run.
- So this is how try & catch shud be used w/ fun if ~~mutl~~ multiple fun call needs to be made.

Restricting a fun from throwing an ex

- A fun can be restricted from throwing certain types of ex.
- This can be done by using a throw stmt after fun name.

eg:- Restrict a fun from throwing certain ex.

$\underbrace{\text{void fun( ) throw (int, char)}}$

fun ↓ fun name

- specify the types of ex that the fun is allowed to throw.
- It can't throw any other ex.

Ex Prog to restrict a fun from throwing any exc other than int & char.

A
```
void fun(int x) throw(int, char)
{
  //this fun can throw only int &
   char ex.
  if (x==0)   throw 100;
  if (x==1)   throw 'a';
  if (x==2)   throw 10.5; //when a
              float ex is thrown it
              will not be handled bcoz
     fun can't throw float ex.

}
int main()
{
      int a;
      try
      {
         fun(0);
         fun(1);
         fun(2); // fun will throw ~~catch~~ float
      }            ex.
catch ( int x)
{
        ≡
 }
 catch ( char x )
 {
        ≡
 }
 catch (float x )
 {
        ≡
 }
}//main ends.
```

- In this prog , fun will throw an ex of type float when fun (2) is called.
- However the fun is restricted from throwing any ex. other than int & char.
- So float ex will not be handled.
- eventhough there is a float catch block but it will not handle float ex thrown by fun.
- So APT will occur.
- Whenever a fun () throws any ex wh. is not allowed than APT will occur.

Rethrowing an Exception

1) An ex can be rethrown.
2) It is done when you want the same ex to be handled by 2 diff handlers in diff ways.
3) One handler will handle certain part of ex & other handler will handle another part of ex.
4) ~~E~~ Ex. can be rethrown by using nested try & catch.
5) The inner catch block will handle some part of ex & it will rethrow the ex.
6) The outer catch block will handle some other part of ex.
7) So 1 ex will be handled 2 times.

eg

```
try
{
        try
        {
                if (a == 0) throw 100;
                // throw an ex.

        }
        catch (int x)
        {
                cout<<" inner catch ";
                throw ; // ** inside inner catch
                                the ex will be
                                rethrown.
        }                   // No value is specified
}
catch (int x)
{           // rethrown ex will be caught by
    cout<<" outer catch";          outer
                                    catch.

}
```