Additional Features

- Catching All Exceptions
- Restricting Exceptions
- Rethrowing an Exception

P Sai First EHM IV
Watch later Share

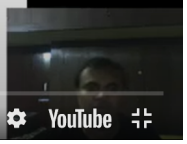
Catching All Exceptions

- Suppose you want exception handler to catch all exceptions instead of just a certain type.
- Then, use

```
catch(...) {
  // process all exceptions
}
```

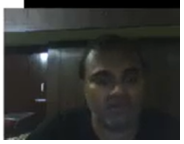
- The ellipsis matches any type of data.

MORE VIDEOS
2:00 / 20:22
YouTube



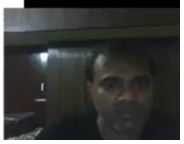
Press Esc to exit full screen

```
#include <iostream>
using namespace std;

void Xhandler(int test)
{
   try
   {
      if(test==0) throw test;
      if(test==1) throw 'a';
      if(test==2) throw 123.23;
   }
   catch(...)
   {
      cout << "Caught One!\n";
   }
}
```



```
#include <iostream>
using namespace std;

void Xhandler(int test)
{
   try
   {
      if(test==0) throw test;
      if(test==1) throw 'a';
      if(test==2) throw 123.23;
   }
   catch(...)
   {
      cout << "Caught One!\n";
   }
}
```

```
int main()
{
   cout << "Start\n";
   Xhandler(0);
   Xhandler(1);
   Xhandler(2);
   cout << "End";
   return 0;
}
```



```
#include <iostream>
using namespace std;
void Xhandler(int test)
{
   try
   {
      if(test==0) throw test;
      if(test==1) throw 'a';
      if(test==2) throw 123.23;
   }
   catch(int i)
   {
      cout << "Caught an integer\n";
   }
   catch(...)
   {
      cout << "Caught One!\n";
   }
}
```

```
int main()
{
   cout << "Start\n";
   Xhandler(0);
   Xhandler(1);
   Xhandler(2);
   cout << "End";
   return 0;
}
```

```
Start
Caught an integer
Caught One!
Caught One!
End
```

Press Esc to exit full screen

P Sai First EHM IV
Watch later Share

Use of catch(...)

- To catch all exceptions that you don't want to handle explicitly.
- By catching all exceptions,
  - Unhandled exception can be prevented from causing an abnormal program termination.

6:09 / 20:22 YouTube

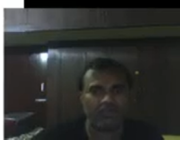
Press Esc to exit full screen

Restricting Exceptions

- We can **restrict the type of exceptions** that a function can throw.
- We can also **prevent a function from throwing** any exceptions whatsoever.

- **The general form**

```
ret-type func-name(arg-list) throw(type-list)
{
  // ...
}
```

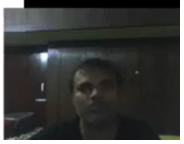


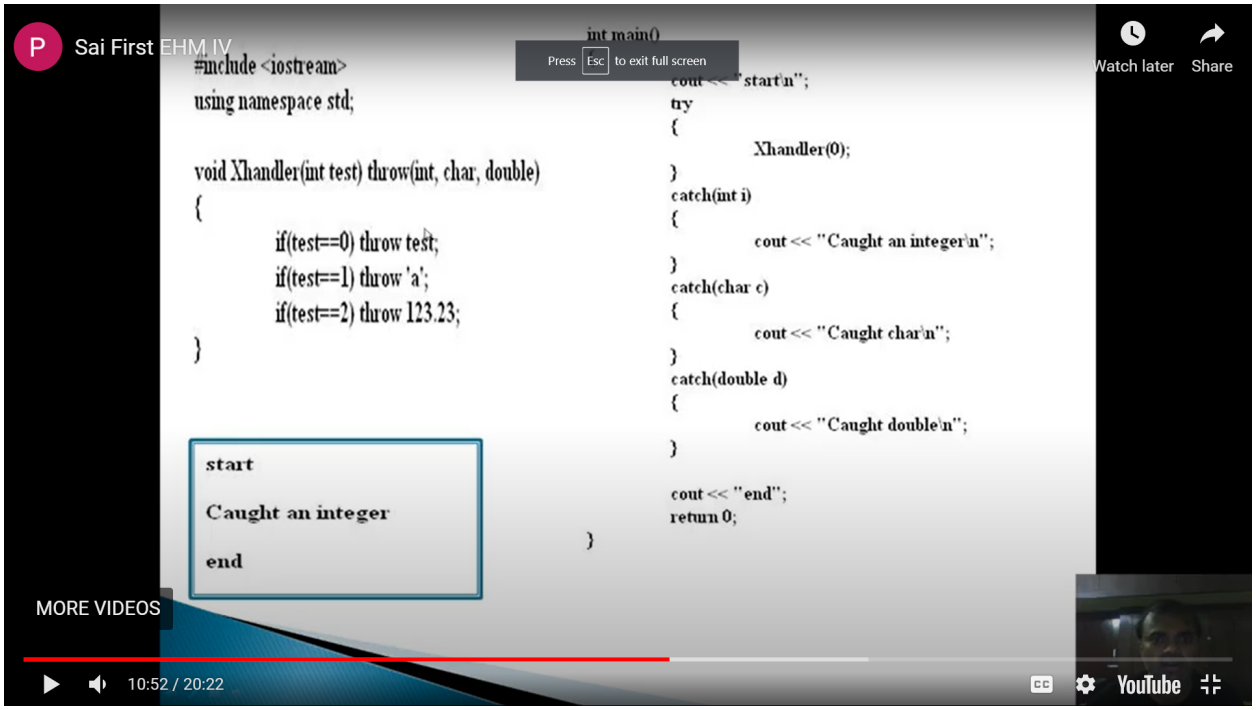
Understanding…

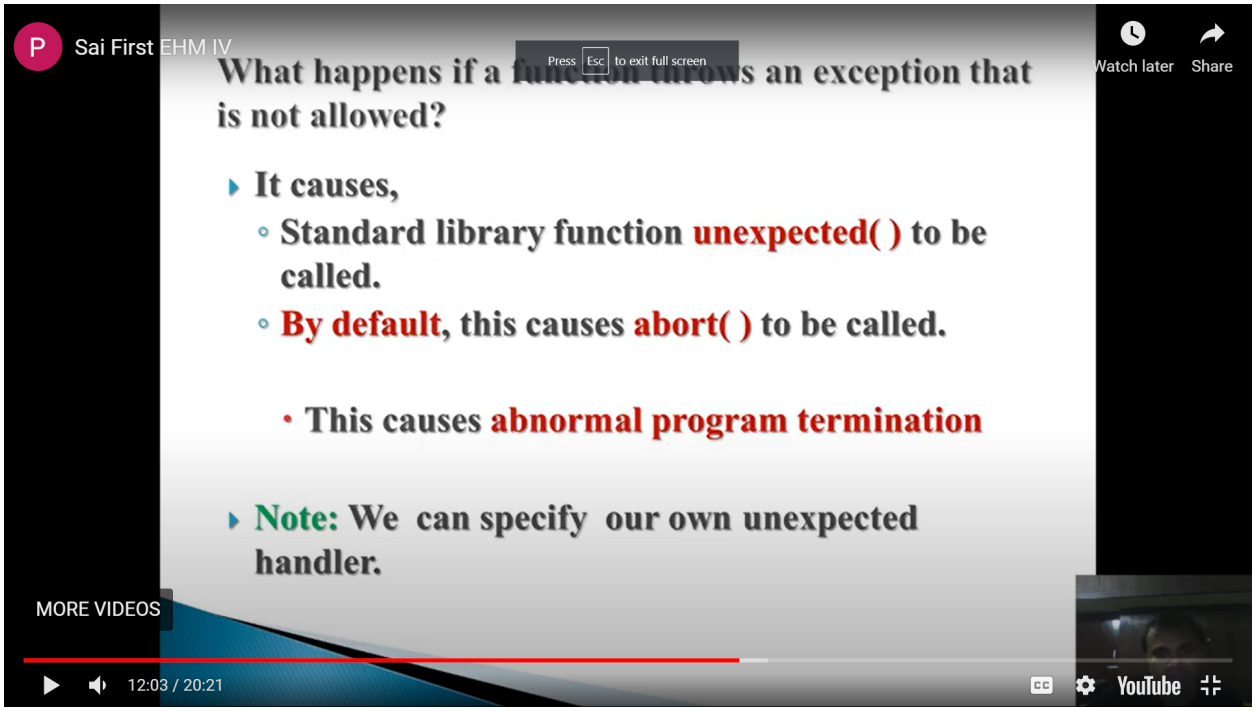
General Form

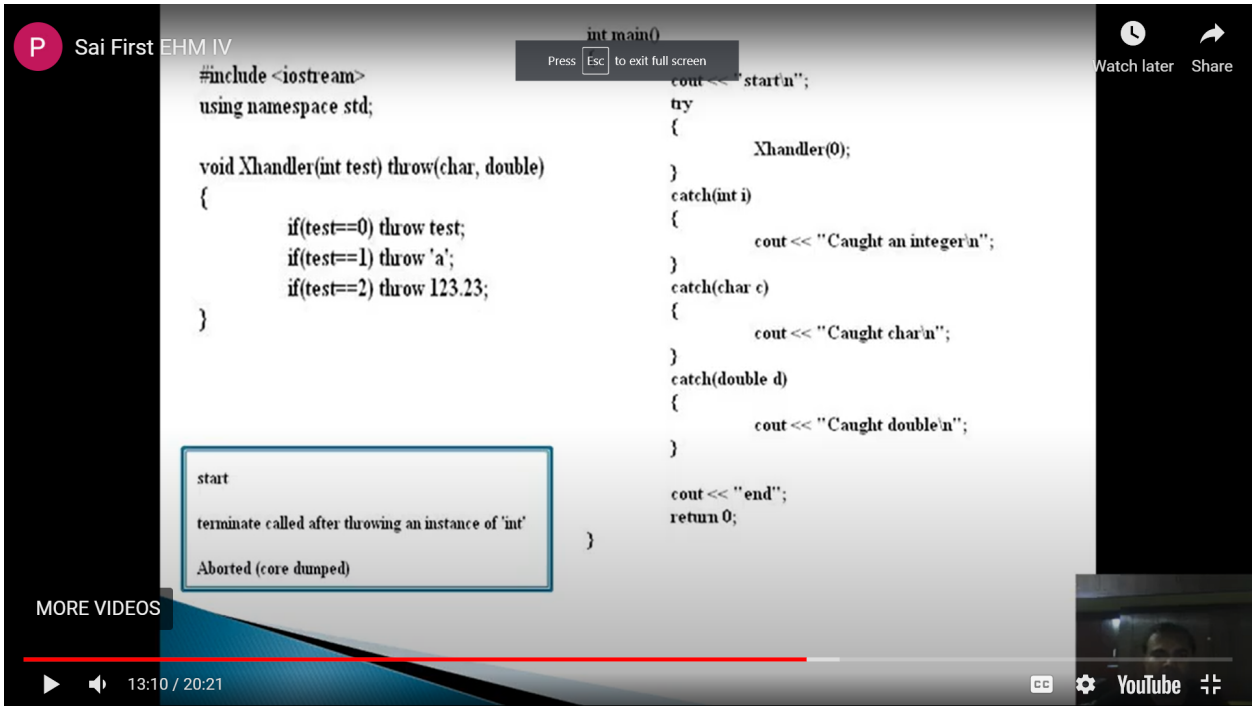
```
ret-type func-name(arg-list) throw(type-list)
{
  // ...
}
```

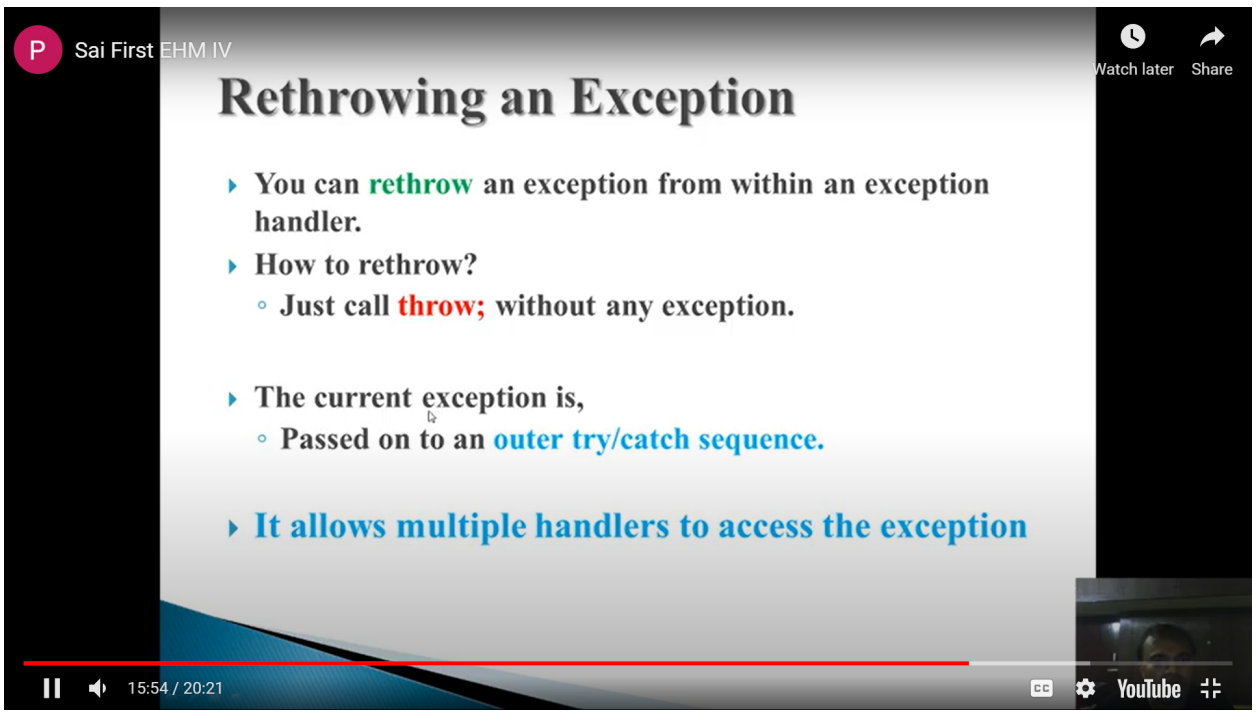
- Only those data types contained in the comma-separated *type-list may be thrown by* the function .
- Throwing any other type of expression will cause **abnormal program termination.**
- If the **list is empty** then, the function is **not allowed to throw any exceptions**.

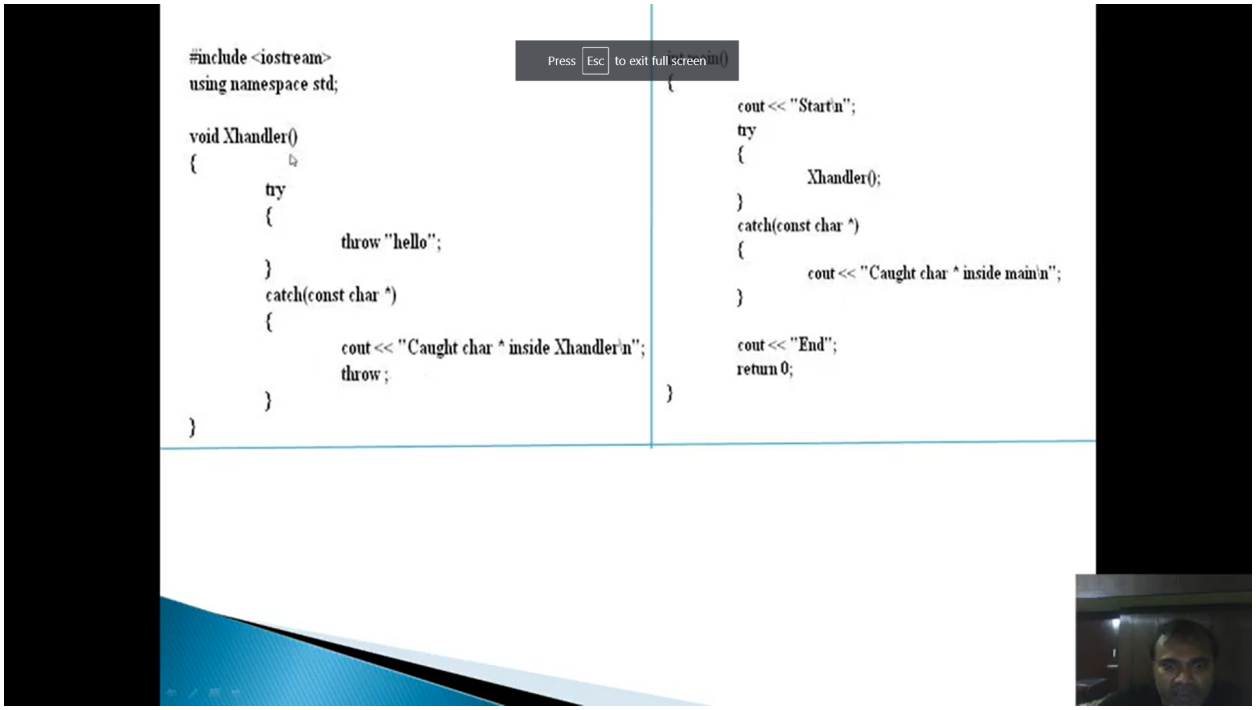



Sai First EHM IV
Press Esc to exit full screen
Watch later
Share
#include <iostream>
using namespace std;
void Xhandler(int test) throw(int, char, double)
{
if(test==0) throw test;
if(test==1) throw 'a';
if(test==2) throw 123.23;
}
int main()
cout << "start\n";
try
{
Xhandler(0);
}
catch(int i)
{
cout << "Caught an integer\n";
}
catch(char c)
{
cout << "Caught char\n";
}
catch(double d)
{
cout << "Caught double\n";
}
cout << "end";
return 0;
}
start
Caught an integer
end
MORE VIDEOS
10:52 / 20:22
YouTube

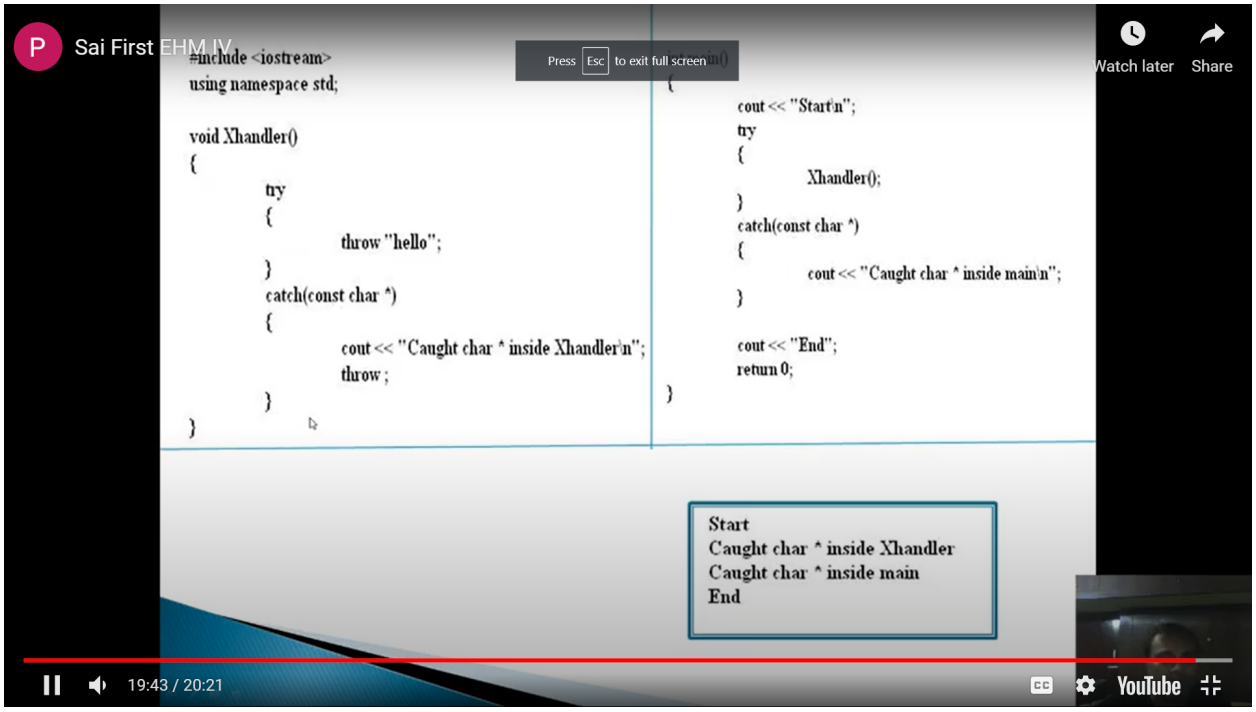

Sai First EHM IV
Press Esc to exit full screen
Watch later
Share
What happens if a function throws an exception that is not allowed?
It causes,
Standard library function unexpected( ) to be called.
By default, this causes abort( ) to be called.
This causes abnormal program termination
Note: We can specify our own unexpected handler.
MORE VIDEOS
12:03 / 20:21
YouTube


Sai First EHM IV
Watch later
Share
Press Esc to exit full screen
#include <iostream>
using namespace std;
void Xhandler(int test) throw(char, double)
{
if(test==0) throw test;
if(test==1) throw 'a';
if(test==2) throw 123.23;
}
int main()
cout << "start\n";
try
{
Xhandler(0);
}
catch(int i)
{
cout << "Caught an integer\n";
}
catch(char c)
{
cout << "Caught char\n";
}
catch(double d)
{
cout << "Caught double\n";
}
cout << "end";
return 0;
}
start
terminate called after throwing an instance of 'int'
Aborted (core dumped)
MORE VIDEOS
13:10 / 20:21
YouTube


Sai First EHM IV
Watch later
Share
Rethrowing an Exception
You can rethrow an exception from within an exception handler.
How to rethrow?
Just call throw; without any exception.
The current exception is,
Passed on to an outer try/catch sequence.
It allows multiple handlers to access the exception
15:54 / 20:21
YouTube


Press Esc to exit full screen
#include <iostream>
using namespace std;
void Xhandler()
{
try
{
throw "hello";
}
catch(const char *)
{
cout << "Caught char * inside Xhandler\n";
throw ;
}
}
{
cout << "Start\n";
try
{
Xhandler();
}
catch(const char *)
{
cout << "Caught char * inside main\n";
}
cout << "End";
return 0;
}

\


Sai First EHM IV
Watch later
Share
Press Esc to exit full screen
#include <iostream>
using namespace std;
void Xhandler()
{
try
{
throw "hello";
}
catch(const char *)
{
cout << "Caught char * inside Xhandler\n";
throw ;
}
}
{
cout << "Start\n";
try
{
Xhandler();
}
catch(const char *)
{
cout << "Caught char * inside main\n";
}
cout << "End";
return 0;
}
Start
Caught char * inside Xhandler
Caught char * inside main
End
19:43 / 20:21
YouTube