

# Try and Catch Blocks Solutions

- Complete the following statement:
  - A try block can be followed by several catch blocks, each of which handles an exception instance with a different (...)
  - **static type**
- If more than one of the catch blocks is able to handle a certain exception, how does the program decide which one to invoke?
  - **It will invoke the first catch block (in source code order) that matches the exception's static type**

- In the following code, which handler is invoked, and why?

```
try {  
    vector<int> v;  
    cout << v.at(2) << endl;           // May throw an exception of type std::out_of_range  
}  
catch (const exception& e) {  
    cout << "std::exception\n";  
}  
catch (const out_of_range& e) {  
    cout << "std::out_of_range\n";  
}
```

- In the following code, which handler is invoked, and why?
  - The exception is handled by the catch block for `std::exception`
  - This is the first catch block that can handle the exception, even though the `std::out_of_range` block is a better match

- When writing an exception handler, how should the exception instance be passed to it? Give a reason for your answer
  - The exception should be caught by reference (to const, unless the handler needs to modify it)
  - Avoids making a copy of the exception object
  - Allows dynamic binding to be used when calling the exception object's member functions
  - Prevents “object slicing” of subclasses

- What guidelines should we follow when writing an exception handler?
  - Keep the code simple
  - Remember the program may not be in a stable state
    - Avoid creating new variables (especially if not built-in types)
    - Avoid allocating memory
    - Avoid calling functions
    - Avoid any code that might throw a fresh exception

- What happens if an exception is thrown in a try block and none of the associated catch blocks can handle it?
  - The program will leave the current scope and look in the enclosing scope for a suitable handler
  - If it does not find one there, it will leave that scope and look in that scope's enclosing scope
  - If the program leaves main()'s scope without finding a suitable handler, the program terminates (by default)

- Write a simple program which throws an exception inside a nested try block, which can only be handled by one of the outer catch blocks
- Write a simple program which calls a function. The function throws an exception inside a try block, which can only be handled by a catch block in the calling function