

# Introduction to Exceptions Solutions

- Write a simple program that throws an exception but does not have any exception-related code
- What happens when this program is run? Explain your results
  - The output is  
terminate called after throwing an instance of 'std::out\_of\_range'
  - An exception is thrown
  - Because there is no code to handle the exception, the program is terminated

- What keyword is used to inform the compiler that some code may throw exceptions?
  - try
- What keyword is used to inform the compiler that some code should be used to handle an exception?
  - catch

- How do we specify which type of exception the code should handle?
  - The type of exception we are going to handle goes in brackets after the catch keyword
- How can dynamic binding be used with exception handlers?
  - Use a reference to the base class as the exception type

- Rewrite your program so that it handles the exception by printing out a message which describes the error condition

- What keyword is used to raise an exception?
  - **throw**
- Write a simple program which raises an exception if the user enters a negative number and handles the exception

- What difference would it make if `vector::at()` returned an error code instead of raising an exception?
- Would any changes be needed to the prototype of `vector::at()`?
  - `vector::at()` already has a return value (the value of the element)
  - One possibility would be to return an `std::pair`, in which the first value is the element value and the second value is the error code

- Write some sample code to show how this would be used

```
auto at_two = vec.at(2);      // Call at() and save result in std::pair
if (at_two.second)           // Element exists
    cout << at_two.first << endl;
else                          // Error
    return at_two.second;

// Now edit the caller to add some code to check for the error
```