CS256

Chapter 16: Exceptions

Hanoi University of Science and Technology

Slide contents are mainly based on the following documents:

- Tony Gaddis, Starting out with C++, Eighth Edition, Pearson, 2015
- Tony Gaddis, Power Points Slides:
  `https://www.umsl.edu/~antognolij/gaddis.html`

Exceptions

Does this function always return the expected value?

```
// An unreliable division function
double divide(int numerator, int denominator)
{
    if (denominator == 0)
    {
        cout << "ERROR: Cannot divide by zero.\n";
        return 0;
    }
    else
        return static_cast<double>(numerator) / denominator;
}
```

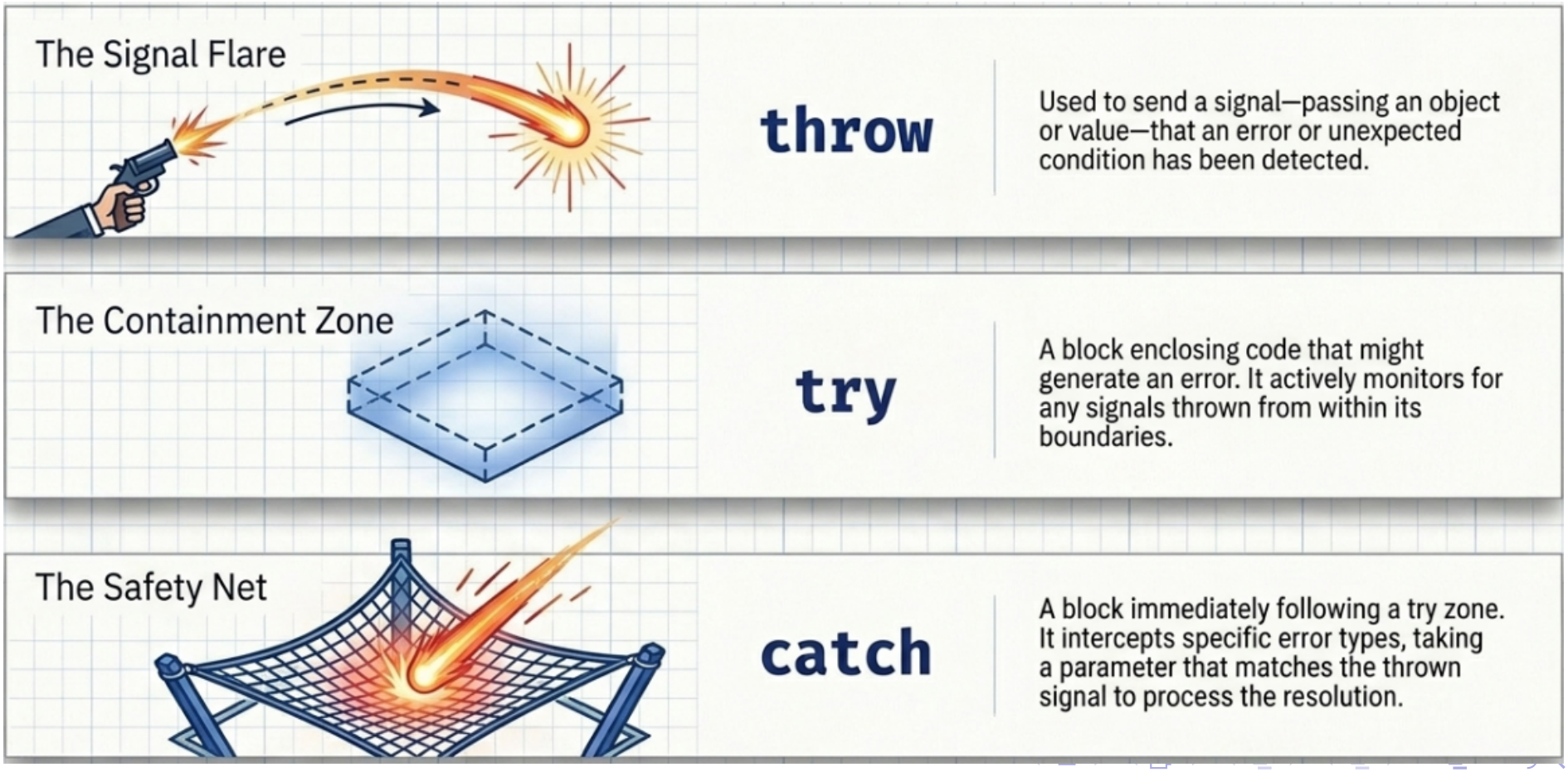
Exceptions

One way of handling complex error conditions is with exceptions:

- Indicate that something unexpected has occurred or been detected.
- Allow program to deal with the problem in a controlled manner
- Can be as simple or complex as program design requires

Exceptions - Terminology

- Exception: object or value that signals an error
- Throw an exception: send a signal that an error has occurred
- Catch/Handle an exception: process the exception; interpret the signal



Exceptions – Flow of Control

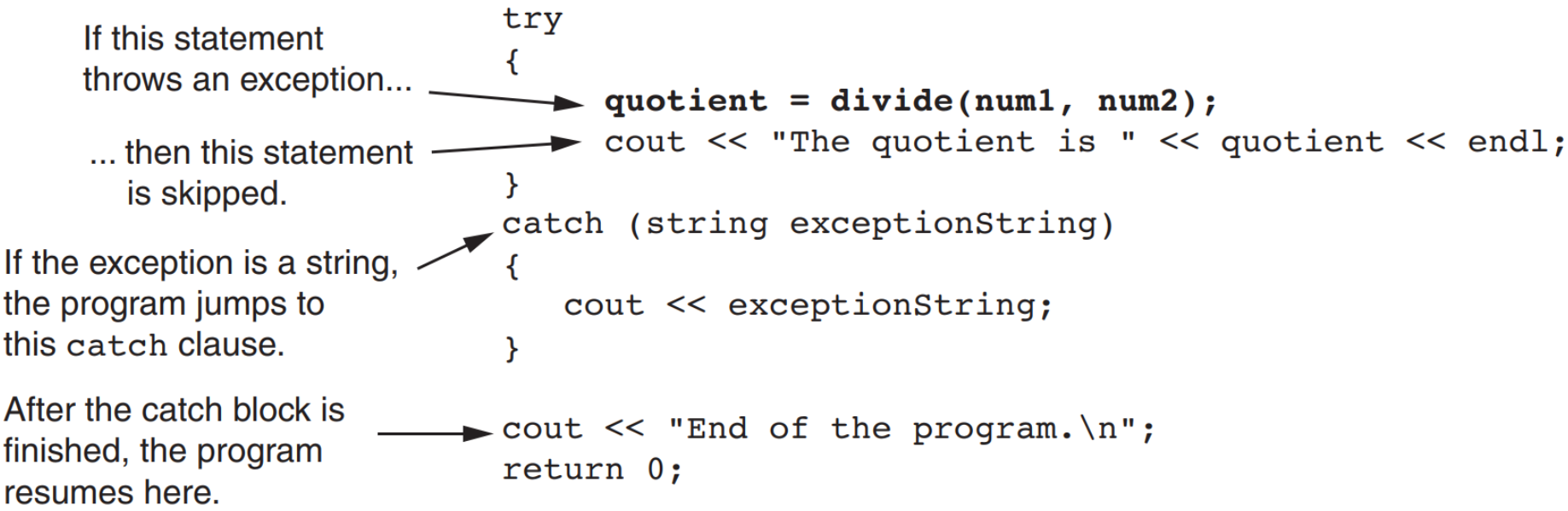
1. A function that throws an exception is called from within a `try` block
2. If the function throws an exception, the function terminates and the try block is immediately exited.
   A `catch` block to process the exception is searched for in the source code immediately following the try block.
3. If a `catch` block is found that matches the exception thrown, it is executed.
   If no catch block that matches the exception is found, the program terminates.

Exceptions – Key Words

```
double divide(int numerator, int denominator)
{
    if (denominator == 0)
        throw string("ERROR: Cannot divide by zero.\n");
    else
        return static_cast<double>(numerator) / denominator;
}
```

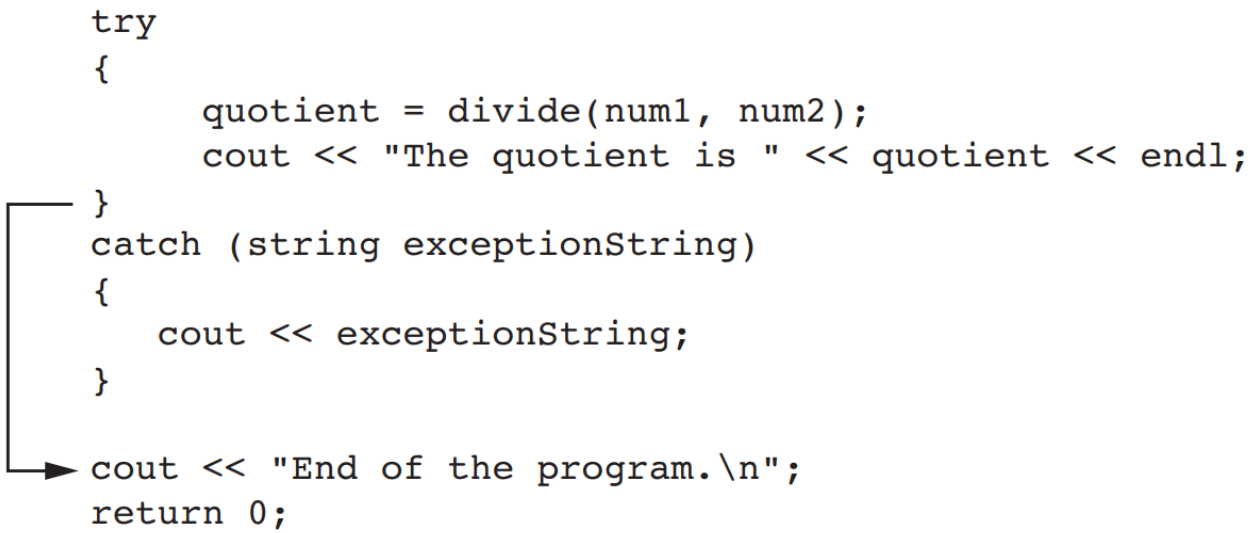
```
try
{
    quotient = divide(num1, num2);
    cout << "The quotient is " << quotient << endl;
}
// catch block handles the specific type of exception thrown
catch (const string& exceptionString)
{
    cout << exceptionString;
}
```

Exceptions - What happens



Exceptions - What happens

If no exception is thrown in the try block, the program jumps to the statement that immediately follows the try/catch construct.



```
try
{
     quotient = divide(num1, num2);
     cout << "The quotient is " << quotient << endl;
}
catch (string exceptionString)
{
   cout << exceptionString;
}

cout << "End of the program.\n";
return 0;
```

Exception Not Caught?

- An exception will not be caught if
  - it is thrown from outside of a `try` block
  - there is no **catch** block that matches the data type of the thrown exception
- If an exception is not caught, the program will terminate

Object-Oriented Exceptions

- An exception class can be defined in a class and thrown as an exception by a member function
- An exception class may have:
  - no members: used only to signal an error
  - members: pass error data to `catch` block
- A class can have more than one exception class
- Example: Program 16.2

Multiple exceptions

```
 // Exception class for a negative width
class NegativeWidth
{ };

// Exception class for a negative length
class NegativeLength
{ };
```

Recover from Errors

Program 16.4

Extracting Data from the Exception Class

```
class NegativeWidth
{
private:
      double value;
public:
      NegativeWidth(double val)
            { value = val; }

      double getValue() const
            { return value; }
};
```

```
catch (Rectangle::NegativeWidth e)
{
      cout << "Error: " << e.getValue()
            << " is an invalid value for the"
            << " rectangle's width.\n";
}
```