CST209

Object-oriented Programming C++

(Week 10)

Prepared by Dr. Teo Bee Guan

Presented by Mr. Venantius Kumar Sevamalai

Content

- Exception Handling

Exception Handling

- One way of handling complex error conditions is with exceptions.
- An exception is a value or an object that signals an error. When the error occurs, an exception is “thrown.”
- Exceptions are used to signal errors or unexpected events that occur while a program is running.

Exception Handling

- For example, the following code shows the divide function, modified to throw an exception when division by zero has been attempted.

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

The throw key word is followed by an argument, which can be any value.

Exception Handling

- To handle an exception, a program must have a try/catch construct.
- The general format of the try/catch construct is:

```
try
{
     // code here calls functions or object member
     // functions that might throw an exception.
}
catch(ExceptionParameter)
{
     // code here handles the exception
}
// Repeat as many catch blocks as needed.
```

In-Class Practice – Example 1

In class Exercise 1

Create a function, calcWages(), to calculate the wages based on the number of working hours. This function should only accept one parameter, hours. The wages is $30 per hour. If the hours is a negative value, the function should throw an exception error.

Test your function by calling your calcWages() function and passing a negative argument.

Object-Oriented Exception Handling with Classes

- Now that you have an idea of how the exception mechanism in C++ works, we will examine an object-oriented approach to exception handling.
- Recall the Rectangle class that was introduced in earlier week. That class had the mutator functions setWidth and setHeight for setting the rectangle's width and length.
- If a negative value was passed to either of these functions, the class displayed an error message and aborted the program.

In-Class Practice – Example 2

In class Exercise 2

Create a class and name it as Triangle. This class should have three private member variables, side1, side2 and side3. Create getter and setter function for each of these member variables and a member function, getPerimeter(), to get the perimeter of the triangle.

In the setter functions, an exception should be thrown if the input parameter value is zero or a negative number.

Test your class in the main function.

Multiple Exceptions

- In many cases a program will need to test for several different types of errors and signal which one has occurred.

- C++ allows you to throw and catch multiple exceptions.

In-Class Practice – Example 3

In class Exercise 3

Modify your previous Triangle class by having multiple exception for each setSide1, setSide2, and setSide3 setter functions.

Test your class in the main function.

Using Exception Handlers to Recover from Errors

- The earlier example of exception handling can have several catch statements to handle different types of exceptions.
- However, the program does not use the exception handlers to recover from any of the errors.
- There is a better way of exception handling by getting valid data from the user to recover from the exceptions.

In-Class Practice – Example 4

In class Exercise 4

Modify your previous example of the main function that implement the Triangle class. The main function should be structured in a way that if either setSide1(), setSide2() or setSide3() throws an exception, the program should prompt the user to enter a valid value.

Extracting Data from the Exception Class

- Sometimes we might want an exception object to pass data back to the exception handler.

- For example, suppose we would like the Rectangle class not only to signal when a negative value has been given, but also to pass the value back.

- This can be accomplished by giving the exception class members in which data can be stored.

In-Class Practice – Example 5

See you next class