Exceptions

Last updated 4/15/20

Exceptions

- Motivation
  - What are we going to do if something breaks?
  - Exception : signals an error or unexpected event during program execution
    - divide by 0
    - type or class error

Exceptions

- Simple – user error exceptions

```
/*
 * simple.cpp
 *
 *  Created on: Apr 24, 2019
 *      Author: johnsontimoj
 */
#include <iostream>
using namespace std;

int main(void){
  int foo;
  foo = 0;
  while((foo < 1) || (foo > 100)){
    cout << "Please enter a value between 1 and 100: ";
    cin >> foo;
  }
  cout << "Thank you, you entered: " << foo << endl;
  return 0;
}
```

```
Please enter a value between 1 and 100: -5
Please enter a value between 1 and 100: 105
Please enter a value between 1 and 100: 5
Thank you, you entered: 5
```

```
Please enter a value between 1 and 100: 3.5
Thank you, you entered: 3
```

```
Please enter a value between 1 and 100: p


                    SYSTEM FREEZES !
```

Exceptions

- Simple – user error exceptions
  - Treat all input as a string – convert to ints

```
/*
 * simple2.cpp
 *
 *  Created on: Apr 24, 2019
 *      Author: johnsontimoj
 */
#include <iostream>
using namespace std;
#include <cstring>

int main(void){
  int foo;
  string foo_str;
  foo = 0;
  while((foo < 1) || (foo > 100)){
    cout << "Please enter a value between 1 and 100: ";
    cin >> foo_str;
    // atoi converts a char array to an int
    // object.c_str converts a c-string to a character array
    foo = atoi(foo_str.c_str());
  }
  cout << "Thank you, you entered: " << foo << endl;
  return 0;
}
```

```
Please enter a value between 1 and 100: -5
Please enter a value between 1 and 100: 105
Please enter a value between 1 and 100: 5
Thank you, you entered: 5
```

```
Please enter a value between 1 and 100: 3.5
Thank you, you entered: 3
```

```
Please enter a value between 1 and 100: p
Please enter a value between 1 and 100: q
Please enter a value between 1 and 100: r
Please enter a value between 1 and 100: this
Please enter a value between 1 and 100: 5
Thank you, you entered: 5
```

Exceptions

- Complex exceptions
  - throw
    - When an error occurs – throw an exception
      - Transfers the program to an “exception handler”
      - The current function is aborted
  - Exception handler
    - try
      - Contains functions or code that might create an exception
    - catch
      - Looks for an exception
      - Executes IFF the correct type of exception is thrown

Exceptions

- Complex exceptions - handler

```
/*
 * throw.cpp
 *
 *  Created on: Apr 25, 2019
 *      Author: johnsontimoj
 */
#include <iostream>
using namespace std;

double divide(int num, int denom);

int main(void){
  int numerator;
  int denominator;
  double result;

  while(1){
    cout << "please enter a numerator and denominator: ";
    cin >> numerator;
    cin >> denominator;
    try{
      result = divide(numerator, denominator);
      cout << "the result is: " << result << endl;
    }
    catch(const char * exception_string){
      cout << exception_string;
    }
  }
}

double divide(int num, int denom){
  if(denom == 0)
    throw "ERROR - Divide by zero - terminating function \n";

  return (static_cast<double>(num) / denom);
}
```

```
please enter a numerator and denominator: 2 3
the result is: 0.666667
please enter a numerator and denominator: 0 3
the result is: 0
please enter a numerator and denominator: 2 0
ERROR - Divide by zero - terminating function
please enter a numerator and denominator: 2 2
the result is: 1
```

code that might cause an exception

what to do if a const char * exception i s thrown

string literal type is const char *

Exceptions

- Complex exceptions – handler failure

```
/*
 * throw.cpp
 *
 *  Created on: Apr 25, 2019
 *      Author: johnsontimoj
 */
#include <iostream>
using namespace std;

double divide(int num, int denom);

int main(void){
  int numerator;
  int denominator;
  double result;

  while(1){
    cout << "please enter a numerator and denominator: ";
    cin >> numerator;
    cin >> denominator;
    try{
      result = divide(numerator, denominator);
      cout << "the result is: " << result << endl;
    }
    catch(int exception_int){
      cout << exception_int;    }
  }
}

double divide(int num, int denom){
  if(denom == 0)
    throw "ERROR - Divide by zero - terminating function \n";

  return (static_cast<double>(num) / denom);
}
```

Since the type of the exception thrown does not match the type of the catch Program aborts

```
please enter a numerator and denominator: 2 3
the result is: 0.666667
please enter a numerator and denominator: 0 3
the result is: 0
please enter a numerator and denominator: 2 0

This application has requested the Runtime to terminate it in an unusual way.
Please contact the application's support team for more information.
```

code that might cause an exception

what to do if an int exception is thrown

string literal type is const char *

Exceptions

- Complex exceptions – handler failure

```
/*
 * throw.cpp
 *
 *  Created on: Apr 25, 2019
 *      Author: johnsontimoj
 */
#include <iostream>
using namespace std;

double divide(int num, int denom);

int main(void){
	int numerator;
	int denominator;
	double result;
	while(1){
		cout << "please enter a numerator and denominator: ";
		cin >> numerator;
		cin >> denominator;
		try{
			result = divide(numerator, denominator);
			cout << "the result is: " << result << endl;
		}
		catch(int exception_int){
		cout << exception_int << endl;
		}
	}
	cout << "end of program";
}

double divide(int num, int denom){
	if(denom == 0)
	    throw 1;
    return (static_cast<double>(num) / denom);
}
```

```
Exceptions.exe [C/C++ Application] D:\GDrive\MSOE\20_Q3_EE2
please enter a numerator and denominator: 2 3
the result is: 0.666667
please enter a numerator and denominator: 0 3
the result is: 0
please enter a numerator and denominator: 2 0
-1
please enter a numerator and denominator: 3 3
the result is: 1
please enter a numerator and denominator:
```

code that might cause an exception

what to do if an int exception is thrown

type is int

Exceptions

- Class / Object exceptions
  - Create an empty class to use as the exception type
    - Throw, try, catch just like before

Exceptions

- Class / Object exceptions

```
/*
 * Rect1.h
 *
 *  Created on: Apr 25, 2019
 *      Author: johnsontimoj
 */

#ifndef RECT1_H_
#define RECT1_H_

class Rect1 {
  private:
    double width;
    double length;

  public:
    // Exceptions
    class NegativeDimension{
      // empty class for exceptions
    };

    // Constructor
    Rect1();

    // Setters
    void setWidth(double w);
    void setLength(double l);

    // getters
    double getWidth(void);
    double getLength(void);

    // calculations
    double calcArea(void);
};

#endif /* RECT1_H_ */
```

Empty class but a default constructor is automatically made

```
/*
 * Rect1.cpp
 *
 *  Created on: Apr 25, 2019
 *      Author: johnsontimoj
 */

#include "Rect1.h"

Rect1::Rect1() {
  setWidth(0.0);
  setLength(0.0);
}

void Rect1::setWidth(double w){
  if(w >= 0)
    width = w;
  else
    throw NegativeDimension();
  return;
}
void Rect1::setLength(double l){
  if(l >= 0)
    length = l;
  else
    throw NegativeDimension();
  return;
}
double Rect1::getWidth(void){
  return width;
}
double Rect1::getLength(void){
  return length;
}
double Rect1::calcArea(void){
  return (width * length);
}
```

calling the default constructor causes a reference to an object of type NegativeDimension to be returned

Exceptions

- Class / Object exceptions

```
/*
 * rect1_test.cpp
 *
 *  Created on: Apr 25, 2019
 *      Author: johnsontimoj
 */

#include "Rect1.h"

#include<iostream>
using namespace std;

int  main(void){
  double  width;
  double length;

  // create a rectangle
  Rect1 myRect;

  // get values
  while(1){
    cout << "Enter the rectangles width and length: ";
    cin >> width >> length;

    //Attempt to update my rectangle
    try{
      myRect.setWidth(width);
      myRect.setLength(length);
      cout << "The area of the rectangle is: " << myRect.calcArea() <<  endl;
    }
    catch(Rect1::NegativeDimension &){
      cout << "ERROR - negative value for width or length" <<  endl;
    }
  }
return 0;
}
```

```
Enter the rectangles width and length: 2 3
The area of the rectangle is: 6
Enter the rectangles width and length: -2 4
ERROR - negative value for width or length
Enter the rectangles width and length: 2 -4
ERROR - negative value for width or length
Enter the rectangles width and length: 2 4
The area of the rectangle is: 8
Enter the rectangles width and length:
```

note: need the fully qualified name

Exceptions

- Class / Object exceptions
  - Can have multiple exception types (classes)

Exceptions

- Class / Object exceptions

```
/*
 * Rect2.h
 *
 *  Created on: Apr 25, 2019
*         Author: johnsontimoj
*/
#ifndef RECT2_H_
#define RECT2_H_

class Rect2 {  private:
double width;
double length;

        public:
// Exceptions
class NegativeWidth{
   // empty class for exceptions
};
class NegativeLength{
// empty class for exceptions
};

// Constructor
Rect2();

// Setters
void setWidth(double w);  void
setLength(double l);

// getters
double getWidth(void);  double
getLength(void);

// calculations
double calcArea(void);

};

#endif /* RECT2_H_ */
```

```
/*
/*  * Rect2.cpp
 *
 *  Created on: Apr 25, 2019
 *      Author: johnsontimoj
 */

#include "Rect2.h"

Rect2::Rect2() {
      setWidth(0.0);
      setLength(0.0);
}

void Rect2::setWidth(double w){
      if(w >= 0)
            width = w;
      else
            throw NegativeWidth();
      return;
}
void Rect2::setLength(double l){
      if(l >= 0)
            length = l;
      else
            throw NegativeLength();
      return;
}
double Rect2::getWidth(void){
      return width;
}
double Rect2::getLength(void){
      return length;
}
double Rect2::calcArea(void){
      return (width * length);
}
```

Exceptions

- Class / Object exceptions

```
/*
 * rect2_test.cpp
 *
 *  Created on: Apr 25, 2019
 *      Author: johnsontimoj
 */

#include "Rect2.h"

#include<iostream>
using namespace std;

int  main(void){
  double  width;
  double length;

  // create a rectangle
  Rect2 myRect;

  // get values
  while(1){
    cout << "Enter the rectangles width and length: ";
    cin >> width >> length;

    //Attempt to update my rectangle
    try{
      myRect.setWidth(width);
      myRect.setLength(length);
      cout << "The area of the rectangle is: " << myRect.calcArea() << endl;
    }
    catch(Rect2::NegativeWidth &){
      cout << "ERROR - negative value for width" << endl;
    }
    catch(Rect2::NegativeLength &){
      cout << "ERROR - negative value for length" << endl;
    }
  }
return 0;
}
```

```
Enter the rectangles width and length: 2 3
The area of the rectangle is: 6
Enter the rectangles width and length: -2 3
ERROR - negative value for width
Enter the rectangles width and length: 2 -3
ERROR - negative value for length
Enter the rectangles width and length: 2 3
The area of the rectangle is: 6
Enter the rectangles width and length:
```

note: need the fully qualified name

Exceptions

- Class / Object exceptions
  - Can attempt to recover from an exception

```
/*
 * rect2_test2.cpp
 *
 *  Created on: Apr 25, 2019
 *      Author: johnsontimoj
 */

#include "Rect2.h"

#include<iostream>
using namespace std;

int main(void){
  double width;
  double length;
  bool try_again;

  // create a rectangle
  Rect2 myRect;

  // get values
  while(1){
    cout << "Enter the rectangles width ";
    cin >> width;
    try_again = true;

    //Attempt to recover if necessary
    while(try_again){
      try{
        myRect.setWidth(width);
        try_again = false;// no exception thrown
      }
      catch(Rect2::NegativeWidth &){
        cout << "Please enter a non-negative value for width: ";
        cin >> width;
      }
    }
    cout << "Enter the rectangles length ";
    cin >> length;
    try_again = true;
    //Attempt to recover if necessary
    while(try_again){
      try{
        myRect.setLength(length);
        try_again = false;// no exception thrown
      }
      catch(Rect2::NegativeLength &){
        cout << "Please enter a non-negative value for length: ";
        cin >> length;
      }
    }
    cout << "The area of the rectangle is: " << myRect.calcArea() << endl;
  }
  return 0;
}
```

```
Enter the rectangles width 2
Enter the rectangles length 3
The area of the rectangle is: 6
Enter the rectangles width -2
Please enter a non-negative value for width: 2
Enter the rectangles length -3
Please enter a non-negative value for length: 3
The area of the rectangle is: 6
```

Exceptions

- Class / Object exceptions
  - Can return information from an exception
  - Provide the exception class with member variables

Exceptions

- Class / Object exceptions
  - Can return information from an exception

```
/*
 * Rect3.h
 *
 *  Created on: Apr 25, 2019
 *      Author: johnsontimoj
 */

#ifndef RECT3_H_
#define RECT3_H_

class Rect3 {
  private:
    double width;
    double length;

  public:
    // Exceptions
    class NegativeWidth{
      private:
        double value;
      public:
        NegativeWidth(double val){
          value = val;
        }
        double getValue(){
          return value;
        }
    };
```

exception classes:
class declaraton has a value
and a way to set/get the value

```
    class NegativeLength{
      private:
        double value;
      public:
        NegativeLength(double val){
          value = val;
        }
        double getValue(){
          return value;
        }
    };

    // Constructor
    Rect3();

    // Setters
    void setWidth(double w);
    void setLength(double l);

    // getters
    double getWidth(void);
    double getLength(void);

    // calculations
    double calcArea(void);

};

#endif /* RECT3_H_ */
```

Exceptions

- Class / Object exceptions
  - Can return information from an exception

```
/*
 * Rect3.cpp
 *
 *  Created on: Apr 25, 2019
 *      Author: johnsontimoj
 */

#include "Rect3.h"

Rect3::Rect3()  {
  setWidth(0.0);
  setLength(0.0);
}

void Rect3::setWidth(double w){
  if(w >= 0)
    width = w;
  else
    throw NegativeWidth(w);
  return;
}

void Rect3::setLength(double l){
  if(l >= 0)
    length = l;
  else
    throw NegativeLength(l);
  return;
}
```

```
double Rect3::getWidth(void){
  return width;
}

double Rect3::getLength(void){
  return length;
}

double Rect3::calcArea(void){
  return (width * length);
}
```

passing values when the exception class is thrown

Exceptions

- Class / Object exceptions
  - Can return information from an exception

```
/*
 * rect3_test.cpp
 *
 *  Created on: Apr 25, 2019
 *      Author: johnsontimoj
 */

#include "Rect3.h"

#include<iostream>
using namespace std;

int main(void){
  double width;
  double length;
  bool try_again;

  // create a rectangle
  Rect3 myRect;

  // get values
  while(1){
    cout << "Enter the rectangles width ";
    cin >> width;
    try_again = true;

    //Attempt to recover if necessary
    while(try_again){
      try{
        myRect.setWidth(width);
        try_again = false;// no exception thrown
      }
      catch(Rect3::NegativeWidth & exptn){
        cout << "You entered " << exptn.getValue() << endl;
        cout << "Please enter a non-negative value for width: ";
        cin >> width;
      }
    }
    cout << "Enter the rectangles length ";
    cin >> length;
    try_again = true;
    //Attempt to recover if necessary
    while(try_again){
      try{
        myRect.setLength(length);
        try_again = false;// no exception thrown
      }
      catch(Rect3::NegativeLength & exptn){
        cout << "You entered " << exptn.getValue() << endl;
        cout << "Please enter a non-negative value for length: ";
        cin >> length;
      }
    }
    cout << "The area of the rectangle is: " << myRect.calcArea() << endl;
  }
  return 0;
}
```

catch – creates an object of type Negative... with name exptn

exptn is used in output

```
Enter the rectangles width 2
Enter the rectangles length 3
The area of the rectangle is: 6
Enter the rectangles width -2
You entered -2
Please enter a non-negative value for width: 2
Enter the rectangles length -3
You entered -3
Please enter a non-negative value for length: 3
The area of the rectangle is: 6
Enter the rectangles width
```