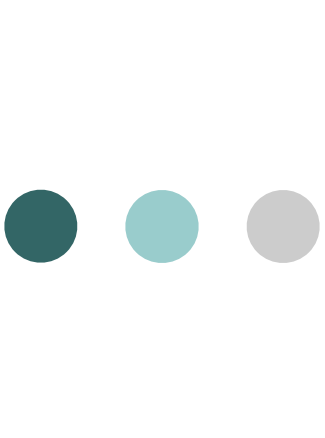
Chapter 16: Exceptions and Templates

Kafi Rahman
Assistant Professor
Truman State University



16.1

Exceptions

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 the 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 – Key Words

- throw – followed by an argument, is used to throw an exception
- try – followed by a block { }, is used to invoke code that may throw an exception
- catch – followed by a block { }, is used to detect and process exceptions thrown in preceding try block.
  - Takes a parameter that matches the type of the exception that may be thrown.

Exceptions – Flow of Control Algorithm

- A function that may throw an exception must be called from within a try block
- 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.
  - 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 – Example (1)

```
// function that throws an exception
int totalDays(int days, int weeks)
{
    if ((days < 0) || (days > 7))
      throw "invalid number of days";

    // the argument to throw is the
    // character string
     else
      return (7 * weeks + days);
}
```

Exceptions – Example (2)

```
int main()
{
    int days = 8, weeks = 10;
    try // block that calls function
    {
        int total_days = totalDays(days, weeks);
        cout << "Total days: " << total_days << endl;
    }
    catch (const char *msg) // interpret exception
    {
        cout << "Error: " << msg << endl;
    }
    cout<<"End of the program" << endl;
    return 0;
}
```

Exceptions – What Happens

- try block is entered. totalDays function is called
- If 1st parameter is between 0 and 7, total number of days is returned and catch block is skipped over (no exception is thrown)
- If exception is thrown,
  - function and try block are exited,
  - catch blocks are scanned for 1st one that matches the data type of the thrown exception.
  - catch block executes

Exceptions: example 16-1

```
 8  int main()
 9  {
10     int num1, num2;  // To hold two numbers
11     double quotient; // To hold the quotient of the numbers
12
13     // Get two numbers.
14     cout << "Enter two numbers: ";
15     cin >> num1 >> num2;
16
17     // Divide num1 by num2 and catch any
18     // potential exceptions.
19     try
20     {
21        quotient = divide(num1, num2);
22        cout << "The quotient is " << quotient << endl;
23     }
24     catch (char *exceptionString)
25     {
26        cout << exceptionString;
27     }
28
29     cout << "End of the program.\n";
30     return 0;
31  }
```

Exceptions: example 16-1

```
33  //*********************************************
34  // The divide function divides numerator by  *
35  // denominator. If denominator is zero, the  *
36  // function throws an exception.             *
37  //*********************************************
38
39  double divide(int numerator, int denominator)
40  {
41     if (denominator == 0)
42        throw "ERROR: Cannot divide by zero.\n";
43
44     return static_cast<double>(numerator) / denominator;
45  }
```

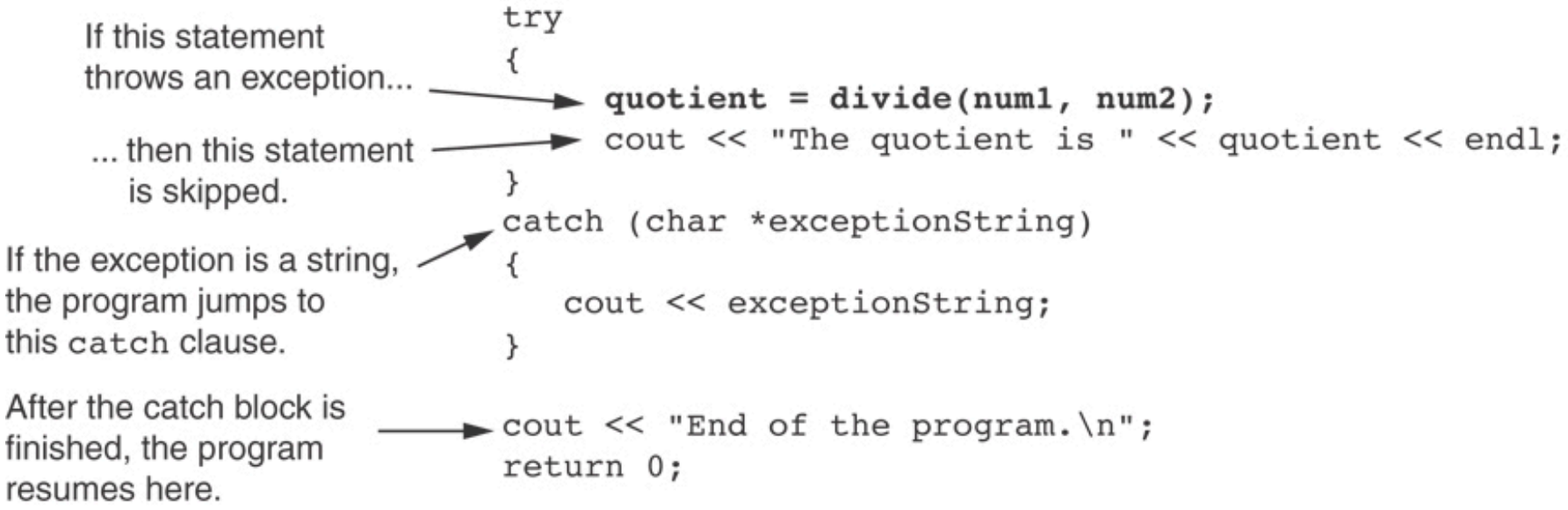
Program Output with Example Input Shown in Bold

```
Enter two numbers: 12 2 [Enter]
The quotient is 6
End of the program.
```

Program Output with Example Input Shown in Bold

```
Enter two numbers: 12 0 [Enter]
ERROR: Cannot divide by zero.
End of the program.
```

What Happens in the Try/ Catch Construct



What if no exception is thrown?

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 (char *exceptionString)
{
   cout << exceptionString;
}

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

General Exception Structure

- When exception occurs, you can throw certain types of values to warn the exception handler
- Depending on the type of values or the received value, the programmer can handle the error in the code

```
1  // exceptions
2  #include <iostream>
3  using namespace std;
4
5  int main () {
6      try
7      {
8          throw 20;
9      }
10     catch (int e)
11     {
12         cout << "An exception occurred."
13              <<" Exception No. " << e << '\n';
14     }
15     return 0;
16 }
```

General Exception Structure: multiple exception types

```
1  // exceptions
2  #include <iostream>
3  using namespace std;
4
5  int main () {
6      try
7      {
8          throw 20;
9      }
10     catch (int e)
11     {
12         cout << "An exception occurred."
13              <<" Exception No. " << e << '\n';
14
15         if(e == 10)
16         { cout<<"\nHandling error "<<e;
17         }
18         else if(e==20)
19         {   cout<<"\nHandling error"<<e;
20         }
21     }
22     return 0;
23 }
```