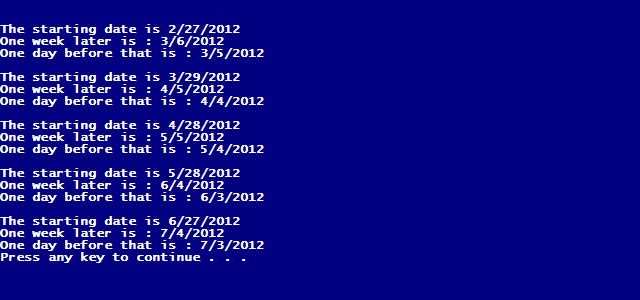
The following are all true about the program listed below:

* Five files ExamTwo.cpp, Date.h,Date.cpp, DateException.h and DateException.cpp are used in compiling and loading this program.
* The Date data type is implemented by the Date class. That class is compiled separately using a Static Library project.
* The DateException data type is implemented by the DateException class. That class is compiled separately using a Static Library project.
* ExamTwo.cpp is a test program that tests the Date data type. It is compiled and run using a Console Application project that links to the object code of the Date and DateException static libraries.
* The strategy of the Date data type is to represent any date as the count of days from January 1st in the year zero. That value is stored as an internal value of type Double within each Date object. This strategy simplifies date comparisons and arithmetic.
* The provided code implements four overloaded operators, +, - , << and >>.
* A Date class variable can be set to a specific number of days for use in arithmetic operations. For example, a Date class variable can be initialized to hold seven days and used to add or subtract a week from another Date class variable.

Here is the output of the test program. You can assume that the program runs correctly.

Here are the listings of the files that contribute to this program.

1. **//---------------------------------------------------------------------------**
2. **// CSC160 Exam Two - ExamTwo.cpp**
3. **//---------------------------------------------------------------------------**
4. **#include <iostream>**
5. **//**
6. **#include "Date.h"**
7. **#include "DateException.h"**
8. **using namespace std;**
9. **using namespace ExamTwo;**
10. **int main()**
11. **{**
12. **try**
13. **{**
14. **Date testOne(2, 27, 2012), testTwo, testThree;**
15. **Date oneWeek(7), oneDay(1);**
16. **for (int count = 0; count < 5; count++)**
17. **{**
18. **try**
19. **{**
20. **cout << "\n\nThe starting date is " << testOne;**
21. **testTwo = testOne + oneWeek;**
22. **cout << "\nOne week later is : " << testTwo;**
23. **testThree = testTwo - oneDay;**
24. **cout << "\nOne day before that is : " << testThree;**
25. **}**
26. **catch (DateException e)**
27. **{**
28. **cout << "Exception: " << e.errorMessage() << endl;**
29. **}**
30. **testOne = testOne + 30;**
31. **}**
32. **}**
33. **catch (DateException e)**
34. **{**
35. **cout << "Exception: " << e.errorMessage() << endl;**
36. **}**
37. **catch (int e)**
38. **{**
39. **cout << "Exception: " << e << endl;**
40. **}**
41. **catch (...)**
42. **{**
43. **cout << "Exception Encountered" << endl;**
44. **}**
45. **cout << endl**
46. **system("PAUSE");**
47. **return 0;**
48. **}**
49. **//---------------------------------------------------------------------------**
50. **// CSC160 Exam Two - Date.h**
51. **//---------------------------------------------------------------------------**
52. **using namespace std;**
53. **namespace ExamTwo**
54. **{**
55. **#ifndef DATE\_H**
56. **#define DATE\_H**
57. **class Date**
58. **{**
59. **public:**
60. **Date();**
61. **Date(int numberOfDays);**
62. **Date(int month, int day, int year);**
63. **void setDate(int month, int day, int year);**
64. **friend Date operator +(const Date& first, const Date& second);**
65. **friend Date operator -(const Date& first, const Date& second);**
66. **friend istream& operator >>(istream& in, Date& thisDate);**
67. **friend ostream& operator <<(ostream& out, const Date& thisDate);**
68. **private:**
69. **double internalDate;**
70. **double numDays(int month, int day, int year);**
71. **void getDetails(int& month, int& day, int& year);**
72. **bool isLeapYear(int year);**
73. **};**
74. **#endif**
75. **}**
76. **//---------------------------------------------------------------------------**
77. **// CSC160 Exam Two - Date.cpp**
78. **//---------------------------------------------------------------------------**
79. **#include <iostream>**
80. **#include "Date.h"**
81. **#include "DateException.h"**
82. **using namespace std;**
83. **namespace ExamTwo**
84. **{**
85. **Date::Date()**
86. **{**
87. **internalDate = 0.0;**
88. **}**
89. **Date::Date(int m, int d, int y)**
90. **{**
91. **internalDate = numDays(m, d, y);**
92. **}**
93. **Date::Date(int days)**
94. **{**
95. **internalDate = static\_cast<double>(days);**
96. **}**
97. **void Date::setDate(int month, int day, int year)**
98. **{**
99. **if (year > 0)**
100. **if (month >= 1 && month <= 12)**
101. **if (day >= 1)**
102. **{**
103. **switch (month)**
104. **{**
105. **case 2:**
106. **if (isLeapYear(year))**
107. **if (day > 29)**
108. **throw DateException("Invalid Day Value: " + day);**
109. **else**
110. **if (day > 28)**
111. **throw DateException("Invalid Day Value: " + day);**
112. **break;**
113. **case 4:**
114. **case 6:**
115. **case 9:**
116. **case 11:**
117. **if (day > 30)**
118. **throw DateException("Invalid Day Value: " + day);**
119. **break;**
120. **default:**
121. **if (day > 31)**
122. **throw DateException("Invalid Day Value: " + day);**
123. **}**
124. **}**
125. **else**
126. **throw DateException("Invalid Day Value: " + day);**
127. **else**
128. **throw DateException("Invalid Month Value: " + month);**
129. **else**
130. **throw DateException("Invalid Year Value: " + year);**
131. **internalDate = numDays(month, day, year);**
132. **}**
133. **double Date::numDays(int month, int day, int year)**
134. **{**

*Code for numDays not shown to save space: you can assume that it works and does not throw any exceptions*

1. **return dayCount;**
2. **}**
3. **void Date::getDetails(int& month, int& day, int& year)**
4. **{**

*Code for getDetails not shown to save space: you can assume that it works and does not throw any exceptions*

1. **}**
2. **bool Date::isLeapYear(int year)**
3. **{**

*Code for isLeapYear not shown to save space: you can assume that it works and does not throw any exceptions*

1. **Date operator +(const Date& first, const Date& second)**
2. **{**
3. **Date temp;**
4. **temp.internalDate = first.internalDate + second.internalDate;**
5. **return temp;**
6. **}**
7. **Date operator -(const Date& first, const Date& second)**
8. **{**
9. **Date temp;**
10. **temp.internalDate = first.internalDate - second.internalDate;**
11. **return temp;**
12. **}**
13. **istream& operator >>(istream& in, Date& thisDate)**
14. **{**
15. **int month, day, year;**
16. **char temp;**
17. **in >> month >> temp >> day >> temp >> year;**
18. **thisDate.internalDate = thisDate.numDays(month, day, year);**
19. **return in;**
20. **}**
21. **ostream& operator <<(ostream& out, const Date& thisDate)**
22. **{**
23. **int month, day, year;**
24. **thisDate.getDetails(month, day, year);**
25. **out << month << '/' << day << '/' << year;**
26. **return out;**
27. **}**
28. **//---------------------------------------------------------------------------**
29. **// CSC160 Exam Two - DateException.h**
30. **//---------------------------------------------------------------------------**
31. **#ifndef DATEEXCEPTION\_H**
32. **#define DATEEXCEPTION\_H**
33. **#include <string>**
34. **using namespace std;**
35. **namespace ExamTwo**
36. **{**
37. **class DateException**
38. **{**
39. **public:**
40. **DateException();**
41. **DateException(string thisErrorMessage);**
42. **string errorMessage();**
43. **private:**
44. **string message;**
45. **};**
46. **}**
47. **#endif**
48. **//---------------------------------------------------------------------------**
49. **// CSC160 Exam Two - DateException.cpp**
50. **//---------------------------------------------------------------------------**
51. **#include <string>**
52. **#include "DateException.h"**
53. **using namespace std;**
54. **namespace ExamTwo**
55. **{**
56. **DateException::DateException()**
57. **{**
58. **message = "Error in Date Occurred: No Additional Information";**
59. **}**
60. **DateException::DateException(string thisErrorMessage)**
61. **{**
62. **message = "Date Exception" + thisErrorMessage;**
63. **}**
64. **string DateException::errorMessage()**
65. **{**
66. **return message;**
67. **}**
68. **}**