EECE.3220: Data Structures

Spring 2019

Key Questions

More C++ basics: output formatting, input functions, strings (Lectures 5-6)

1. Explain the use of `setprecision`. Why is `fixed` necessary?
2. Explain the stream manipulator `showpoint`.
3. Explain the function used to input one or more characters, including whitespace.
4. Explain the function used to input an entire line. What issues exist when mixing this function with the stream extraction operator? (>>) How can we fix those issues?
5. Explain how comparison operators (==, !=, <, >, <=, >=) can be used to compare strings.
6. Explain how string concatenation works in C++.
7. Explain the operation of the `substr()` function.
8. Explain how to access individual characters within a string.

EXAMPLES

1. What is the output of the following program?

```
#include <iostream>
#include <iomanip>
#include <cmath>
using namespace std;

int main()
{
   double root2 = sqrt( 2.0 ); // calc square root of 2
   int places;                 // precision, vary from 0-9
   cout << "Square root of 2 with precisions 0-9." << endl;

   cout << fixed; // use fixed point format (not sci. not)

   // set precision for each digit, then show square root
   for ( places = 0; places <= 9; places++ )
      cout << setprecision( places ) << root2 << endl;
   return 0;
}
```

2. Show the output of the program below if the input stream is:
   1 2 3.4 5

```
// NOTE: The include and using statements are not shown
int main()
{
   double i, j, x, y;
   cin >> i >> j >> x >> y;
   cout << fixed << showpoint;
   cout << "First output " << endl;
   cout << i << ',' << j << ','
       << setprecision(3) << x << ',' << y << endl;
   return 0;
}
```

3. List the output for each of the following code snippets from the same program.

```
int main()
{
   string s1( "happy" );
   string s2( " birthday" );
   string s3;
   // test overloaded equality and relational operators
   cout << "s1 is \"" << s1 << "\"; s2 is \"" << s2
      << "\"; s3 is \"" << s3 << '\"'
      << "\n\nThe results of comparing s2 and s1:"
      << "\ns2 == s1 yields " << ( s2 == s1 ? "true" : "false" )
      << "\ns2 != s1 yields " << ( s2 != s1 ? "true" : "false" )
      << "\ns2 >  s1 yields " << ( s2 > s1 ? "true" : "false" )
      << "\ns2 <  s1 yields " << ( s2 < s1 ? "true" : "false" )
      << "\ns2 >= s1 yields " << ( s2 >= s1 ? "true" : "false" )
      << "\ns2 <= s1 yields " << ( s2 <= s1 ? "true" : "false" );
```

OUTPUT:

```
// test string member function empty

   cout << "\n\nTesting s3.empty():" << endl;
   if ( s3.empty() )
   {
      cout << "s3 is empty; assigning s1 to s3;" << endl;
      s3 = s1; // assign s1 to s3
      cout << "s3 is \"" << s3 << "\"";
   } // end if
   // test overloaded string concatenation operator
   cout << "\n\ns1 += s2 yields s1 = ";
   s1 += s2; // test overloaded concatenation
   cout << s1;
   // test concatenation operator with C-style string
   cout << "\n\ns1 += \" to you\" yields" << endl;
   s1 += " to you";
   cout << "s1 = " << s1 << "\n\n";
```

OUTPUT:

```
// test string member function substr

   cout << "The substring of s1 starting at location 0 for\n"
      << "14 characters, s1.substr(0, 14), is:\n"
      << s1.substr( 0, 14 ) << "\n\n";
   // test substr "to-end-of-string" option
   cout << "The substring of s1 starting at\n"
      << "location 15, s1.substr(15), is:\n"
      << s1.substr( 15 ) << endl;
   // test using subscript operator to create lvalue
   s1[ 0 ] = 'H';
   s1[ 6 ] = 'B';
   cout << "\ns1 after s1[0] = 'H' and s1[6] = 'B' is: "
      << s1 << "\n\n";
   // test subscript out of range with string member function "at"
   cout << "Attempt to assign 'd' to s1.at( 30 ) yields:" << endl;
   s1.at( 30 ) = 'd'; // ERROR: subscript out of range
   return 0;
} // end main
```

OUTPUT: