Ch04-LibraryAndFunction

September 16, 2020

1 Library and Function

1.1 Topics

- some common C++ libraries and how to use them
- iostream, string, numerics, iomanip, cmath, stdlib, sstream, etc.

1.2 Library

- C++ provides a rich set of standard libraries: https://en.cppreference.com/w/cpp/header
- collection of code base that perform various generic and common tasks
  - e.g., input and output, basic math, output formatting, networking and communications, etc.
- C++ program can also use C libraries
- there are other third party libraries as well
  - e.g., boost (https://www.boost.org/) - usable across broad spectrum of applications
  - googletest (https://github.com/google/googletest) - unittest framework by Google
- syntax to include library in your C++ source files is:

```
#include <libraryName>
```

- C-libraries have `c` prefix before the library name

  - e.g., cstdio, cmath, etc.

- one can then use the identifiers (typically, functions, operators, and data) defined in the library

- we'll next dive into some libraries and their functions

1.3 `<iostream>` library

- we've already used **<iostream>** library and some of its functionalities
- iostream provides identifiers such as cin, cout, endl, etc. that aid in standard input/ouput (IO)

```
[1]: // standard input example
     #include <iostream>
     using namespace std;
```

```
[2]: cout << "Hello World!" << endl;
```

```
Hello World!
```

[3]:
```cpp
// standard output example
float num;
cout << "enter a number: ";
cin >> num;
cout << "you entered " << num << endl;
```

```
enter a number: 9.99
you entered 9.99
```

1.4 `<string>` library

- provides string advanced data types
- we used to_string( ) defined to convert numeric data to string type
- there's a lot of other methods provided in string objects
    - we'll dive into this later in string chapter
- there's also a `<cstring>` library completely different from C++ `<string>` library
    - http://www.cplusplus.com/reference/cstring/
    - provides functions for working with c-string (array of char)

[4]:
```cpp
#include <string>
using namespace std;
```

[5]:
```cpp
string some_name = "John Smith";
// convert float to string
string value = to_string(5324.454);
```

[6]:
```cpp
cout << some_name << " " << value << endl;
```

```
John Smith 5324.454000
```

[7]:
```cpp
value
```

[7]: "5324.454000"

[8]:
```cpp
// convert integer to string
string str_num = to_string(234);
```

[10]:
```cpp
// example of c-string (array of characters)
char richest_person[] = "Bill Gates";
```

[11]:
```cpp
richest_person
```

[11]: "Bill Gates"

[12]:
```cpp
// convert c-string to C++ string
string some_name1 = string(richest_person);
```

1.5 `<cstdlib>` library

- provies a bunch of typcasting functions
- orginially in the C standard library as **<stdlib.h>** https://en.cppreference.com/w/cpp/header/cstdlib
- must include **<cstdlib>**
- `float( )`, `int( )`, `double( )`, `char( )` are built-in functions used to convert data types
- `atof( )` - converts a byte string to a floating point value
- `atoi( )`, `atol( )`, `atoll( )` - convert a byte string to corresponding integer value
- the value in parenthesis is called the **argument**

```
[13]: #include <cstdlib> // or
      // include <stdlib.h>

      cout << float(25) << " " << double(20.99f) << " " << int('A') << " " <<␣
        →char(97) << endl;
```

```
25 20.99 65 a
```

```
[14]: cout << atoi("99.99") << " " << atof("89.99");
```

```
99 89.99
```

```
[15]: // generate random number between 0 and RAND_MAX
      // run this cell many times to see different pseudo random number
      rand()
```

```
[15]: 1441282327
```

```
[16]: RAND_MAX
```

```
[16]: 2147483647
```

```
[17]: // generate a random number between 1 and 1000
      rand()%1000+1
```

```
[17]: 730
```

1.6 Numerics library

- https://en.cppreference.com/w/cpp/numeric
- includes common mathematical functions and types
- we may be familiar with a lot of math functionalities from trignometrics or algebra
- expressions such as $sin(\frac{\pi}{2}), log(\frac{1}{x})$, etc.
  - first, we evaluation the expression inside the parenthesis called **argument**
  - then, we apply the function to evaluate the answer

1.7 `<cmath>` library

- provies functionalities to calculate common mathematical expressions
- abs( ), sqrt( ), sin( ), cos( ), pow( ), sqrt( ), log( ), etc.
- more: https://en.cppreference.com/w/cpp/numeric

```cpp
[18]: #include <cmath>
      #include <iostream>

      using namespace std;
```

```cpp
[59]: // can use built-in macro M_PI for the value of M_PI
      M_PI
```

```
[59]: 3.1415927
```

```cpp
[19]: // sine of (pi/2)
      sin(3.141592653589793238/2)
```

```
[19]: 1.0000000
```

```cpp
[20]: cos(0)
```

```
[20]: 1.0000000
```

```cpp
[21]: int x;
```

```cpp
[22]: cout << "Enter a number: ";
      cin >> x;
      cout << "natural ln (" << x << ") = " << log(x); // returns natural log base e
```

```
Enter a number:


10
natural ln (10) = 2.30259
```

```
[22]: @0x10cd5fec0
```

```cpp
[23]: cout << "base 2 log: log2(" << x << ") = " << log2(x); // returns base 2 log
```

```
base 2 log: log2(10) = 3.32193
```

```
[23]: @0x10cd5fec0
```

```cpp
[24]: cout << "base 10 log: log10(" << x << ") = " << log10(x); // returns base 10 log
```

```
base 10 log: log10(10) = 1
```

4

[24]: @0x10cd5fec0

[25]:
```
pow(2, 4) // returns x^y
```

[25]: 16.000000

[26]:
```
sqrt(100) // returns  square root of x
```

[26]: 10.000000

[27]:
```
cbrt(1000) // returns cubic root of x
```

[27]: 10.000000

[28]:
```
// returns absolute positive value of an integer
abs(-7)
```

[28]: 7

[29]:
```
// returns rounded up integer
ceil(5.1)
```

[29]: 6.0000000

[30]:
```
// returns the rounded down integer
floor(5.9)
```

[30]: 5.0000000

[31]:
```
// returns the smallest integer larger than argument
ceil(-5.1)
```

[31]: -5.0000000

[32]:
```
// returns the largest integer smaller than argument
floor(-5.9)
```

[32]: -6.0000000

1.8 `<cctype>` library

- C library that provides some functionalities to work with character types
- `tolower(x)` : returns the lowercase ASCII value of x character
- `toupper(x)` : returns the uppercase of x character
- `isalpha(x)` : checks if a character is alphabetic
- more on cctype: https://en.cppreference.com/w/cpp/header/cctype

```
[33]: #include <cctype>
      using namespace std;
```

```
[34]: tolower('A')
```

[34]: 97

```
[35]: tolower('$')
```

[35]: 36

```
[36]: // convert lowercase ASCII value to char
      char(tolower('A'))
```

[36]: 'a'

```
[37]: char(toupper('z'))
```

[37]: 'Z'

```
[38]: char(toupper('1'))
```

[38]: '1'

```
[39]: // return 1 for true
      isalpha('q')
```

[39]: 1

```
[40]: // returns 0 for false
      isalpha('*')
```

[40]: 0

```
[41]: // TODO: practice with other functions in cctype
```

1.9 `<sstream>` library

- provides high-level string input/output operations
- there are two string stream types (input and output)
- `basic_istringstream` provides functionalities for high-level string stream input operations
  - helps parse string data and extract values as specific types
- `basic_ostringstream` provides functionalities for high-level string stream output operations
  - helpful in collecting results of different data types
- more: https://en.cppreference.com/w/cpp/header/sstream

```
[9]:  #include <sstream> // istringstream and ostringstream
      #include <iostream>
      #include <string>

      using namespace std;
```

```
[10]: // let's say we've a string data record as: firstName MI lastName age GPA
      string mixedData = "John B Doe 20 3.9";
      // let's parse it using istringstream
      istringstream iss(mixedData);
      // now since we created input string stream, iss, we can extract data from it
      // as if we're extracting from standard input stream
```

```
[11]: // let's declare variables to store data into
      string firstName, lastName;
      char MI;
      int age;
      float GPA;
```

```
[12]: iss >> firstName >> MI >> lastName >> age >> GPA;
```

```
[16]: cout << "Student: " << lastName << ", " << firstName << " Age: " << age << "␣
      ↪GPA: " << GPA;
```

    Student: Doe, John Age: 20 GPA: 3.900000

```
[47]: // let's declare an empty output string stream
      ostringstream outstream;
```

```
[48]: // let's write data to outstream just like writing to std output stream
      outstream << firstName << MI << lastName << age << GPA;
```

```
[49]: // let's print the outstream as string
      cout << outstream.str();
      // many objects have methods that can be invoked using . operator
```

    JohnBDoe203.9

[49]: @0x10cd5fec0

1.10 `<iomanip>` library

- provides functionalities to manipulate or format input and output
- `setfill(char)` - changes the fill character; used in conjunction with setw( )
- `setprecision(int)` - changes floating-point precision
- `setw(int)` - changes the width of the next input/output field
- more: https://en.cppreference.com/w/cpp/header/iomanip
- syntax:

```
cout << expression << manipulator << expression << manipulator << ...;
```

- some other manipulators are
  - **fixed** - output the floating point in fixed decimal format
  - **showpoint** - displays the trailing zeros
- parameterized manipulator – the ones with ( ) – require **iomanip** library
- manipulators without parameters require **iostream** library

1.10.1   Tabular output

- often you have to format your output to look well organized
  - like a tabular report
- let's print the following output

```
========================================
First Name       Last Name       Age    GPA
========================================
John             Smith            20     3.9
Alice            Wonderland       19     4.0
****************************************
```

- First Name has width of 20 characters
- Last Name has width of 20 chars
- Age has width of 5 chars
- GPA has width of 5 chars

[2]:
```
#include <iomanip>
#include <iostream>

using namespace std;
```

[51]:
```
// setw() and setfill() example
// print 50 character long string with '='
cout << setw(50) << setfill('=') << "";
```

```
==================================================
```

[52]:
```
cout << setw(20) << "First Name" << setw(20) << "Last Name"
     << setw(5) << "Age" << setw(5) << "GPA" << endl;
// by default data in setw() column is right algined!
```

```
==========First Name===========Last Name==Age==GPA
```

[52]: @0x10cd5fec0

[53]:
```
// let's left align the first name and last name columns
// let's right align the Age and GPA numeric columns
cout << setfill(' '); // need to reset the fill character to ' ' space
cout << setw(20) << left << "First Name" << setw(20) << "Last Name"
     << right << setw(5) << "Age" << setw(5) << "GPA" << endl;
```

```
// by default data in setw() column is right algined!
```

```
First Name          Last Name               Age  GPA
```

[53]: @0x10cd5fec0

[54]:
```
// outputting floating point numbers
cout << 1.234567 << endl;
cout << 1.00000 << endl;
// rounds to 5 decimal points or ignores trailing 0s
```

```
1.23457
1
```

[54]: @0x10cd5fec0

[3]:
```
// force trailing zeros to display
cout << fixed << showpoint << 1.000000 << endl;
```

```
1.000000
```

[4]:
```
// we can fix this by forcing floating point numbers to print using fixed␣
 ↪format
// and then setting the precision
cout << fixed << setprecision(6) << 1.123456789 << " " << 1.000000000 << endl;
```

```
1.123457 1.000000
```

[4]: @0x1089e0ed0

[7]:
```
// Note: fixed and setprecision() manipulators apply to all the floating points␣
 ↪printed subsequently...
cout << 1.0 << '\t' << 9.99 << endl;
```

```
1.000000        9.990000
```

[57]:
```
// let's put it all together
cout << setw(50) << setfill('=') << "" << endl;
cout << setfill(' '); // need to reset the fill character to ' ' space
cout << setw(20) << left << "First Name" << setw(20) << "Last Name"
     << right << setw(5) << "Age" << setw(5) << "GPA" << endl;
cout << setw(50) << setfill('=') << "" << endl;
cout << setfill(' ') << fixed << setprecision(1);
cout << setw(20) << left << "John" << setw(20) << "Smith"
     << right << setw(5) << 20 << setw(5) << 3.9 << endl;
cout << setw(20) << left << "Alice" << setw(20) << "Wonderland"
     << right << setw(5) << 19 << setw(5) << 4.0 << endl;
cout << setw(50) << setfill('*') << "" << endl;
```

```
==================================================
First Name          Last Name           Age  GPA
==================================================

John                Smith                20  3.9
Alice               Wonderland           19  4.0
**************************************************
```

1.11 `<cstdio>` library

- C alternative to C++ iostream library is worth learning about
- C library for stdio has many functions for working with standard input/output
    – https://en.cppreference.com/w/cpp/header/cstdio
- specially, `printf()` can be very useful in quickly printing integers and floaing point numbers with some formatting
- printf function prototype:

```cpp
int printf(const char* format, ...);
```

- format strings include format parameter with **%** symbol to format the given data with
- NOTE: `printf( )` is not supported with C++ Jupyter Notebook kernel
- see examples here: demo_programs/Ch04/printf/printf_demo.cpp

1.12 Exercises

1. Area and perimeter of rectangle
    - Write a C++ program with alogrithm that prompts user to enter length and width of a rectangle. Program then computes its area and perimeter and displays the results.
    - Use as many libraries as possible!
    - see this sample solution exercises/Ch04/rectangle/main.cpp
2. Area and perimeter of a triangle
    - Write a C++ program with alogrithm that prompts user to enter three sides of a triangle. Program then computes its area and perimeter and displays the results.
    - Hint: Use   Heron's   formula: https://www.mathsisfun.com/geometry/herons-formula.html
    - Use as many libraries as possible!
3. Area and volume of a right cylinder
    - Write a C++ program with algorithm that prompts user to enter radius and height of a cyliner. Program then computes and displays the area and volume.
    - Use as many libraries as possible (more the better!)
    - perimeter formula by Google
    - area formula by Google
    - volume formula by Google
4. Area and perimeter of a regular hexagon
    - Write a C++ program with algorithm that prompts user to enter a side of the regular hexagon. It then computes and prints the area and perimeter.
    - area of a regular hexagon by Google
    - Use as many libraries as possible!
5. Average grade
    - Write a C++ program with algorithm that prompts user to enter a student's full name

and three test scores in on line. Program then finds the average score and displays the results as a tabular report.

- must use sstream library to read and write data.
- use as many other libraries as possible.
- e.g., for the sample input: John C Doe 100 95 98
- output should look like the following:

```
----------------------------------------------------------------
First Name    MI    Last Name     Test1   Test2   Test3   Avg.
################################################################
John          C.    Doe             100      95      98  97.66
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
```

1.13   Kattis Problems

- almost every Kattis problem requires at least **iostream** or **cstdio** library (cin or cout)
- math problems may require **cmath** library
- string problems may require **string** library
- **iomanip** is required if output results need to be formatted in certain way
- **cctype** is required by any problem that needs to work with **char** type
- **cstdlib** has many utility functions that may also be required

1.14   Summary

- learned about some common libraries
- purpose of libraries and example usages
- revisited iostream, string, stdlib, cctype, etc. libraries
- learned about cmath, sstream, iomanip with some examples
- exercises and sample solutions

[ ]: