Standard Libraries*

Kenneth Leroy Busbee

Abstract

A general discussion of Standard Libraries with a demonstration program in C++ and an attachment containing other standard C++ library information.

1 Overview of Standard Libraries

Many common or **standard functions**, whose definitions have been written, are ready to be used in any program. They are organized into a group of functions (think of them as several books) and are collectively called a **Standard Library** There are many function organized into several libraries For example, within C++ many math functions exist and have been coded (and placed into libraries). These functions were written by programmers and tested to insure that they work properly. In most cases the functions were reviewed by several people to double and triple check to insure that they did what was expected. We have the advantage of using these functions with **confidence** that they will work properly in our programs, thus saving us time and money.

A main program must establish the existence of functions used in that program. Depending on the programming language, there is a formal way to:

1. define a function
2. declare a function (a prototype is a declaration to a compiler)
3. call a function

When we create functions in our program, we usually see them in the following order in our source code listing:

1. declare the function (prototype)
2. call the function
3. define the function

When we use functions created by others that have been organized into library, we include a header file in our program which contains the prototypes for the functions. Just like functions that we create, we see them in the following order in our source code listing:

1. declaring the function (prototype provided in the include file)
2. call the function (with parameter passing of values)

*Version 1.6: Mar 20, 2010 4:47 pm -0500
†http://creativecommons.org/licenses/by/2.0/

3. define the function (it is either defined in the header file or the linker program provides the actual object code from a Standard Library object area)

In most cases, the user can look at the prototype and understand exactly how the communications (parameter passing) into and out of the function will occur when the function is called. Let's look at the math example of absolute value. The prototype is:

```
int abs(int number);
```

Not wanting to have a long function name the designers named it: **abs** instead of "absolute". This might seem to violate the identifier naming rule of using meaningful names, however when identifier names are established for standard libraries they are often shortened to a name that is easily understood by all who would be using them. The function is of data type int, meaning that the function will return an integer value. It is obvious that the integer value returned is the answer to the question, "What is the absolute value of the integer that is being passed into the function". This function is passed only one value; an int number. If I had two integer variables named apple and banana; and I wanted to store the absolute value of banana into apple; then a line of code to call this function would be:

```
apple = abs(banana);
```

Let's say it in English, pass the function absolute the value stored in variable banana and assign the returning value from the function to the variable apple. Thus, if you know the prototype you can usually properly call the function and use its returning value (if it has one) without ever seeing the definition of the code (i.e. the source code that tells the function how to get the answer; that is written by someone else; and either included in the header file or compiled and placed into an object library; and linked during the linking step of the Integrated Development Environment (IDE).

2 Demonstration Program in C++

2.1 Creating a Folder or Sub-Folder for Source Code Files

Depending on your compiler/IDE, you should decide where to download and store source code files for processing. Prudence dictates that you create these folders as needed prior to downloading source code files. A suggested sub-folder for the **Bloodshed Dev-C++ 5 compiler/IDE** might be named:

- Demo_Programs

If you have not done so, please create the folder(s) and/or sub-folder(s) as appropriate.

2.2 Download the Demo Program

Download and store the following file(s) to your storage device in the appropriate folder(s). You may need to right click on some of the links and select "Save Target As" in order to download some of the files. Following the methods of your compiler/IDE, compile and run the program(s). Study the source code and/or other file(s) in conjunction with other learning materials.

Download from Connexions: Demo_Standard_Libraries.cpp[1]

Download from Connexions: Demo_Standard_Libraries_Listing.txt[2]

3 Definitions

Definition 1: Standard Library

A set of specific task functions that have been added to the programming language for universal use.

[1] See the file at <http://cnx.org/content/m19202/latest/Demo_Standard_Libraries.cpp>

[2] See the file at <http://cnx.org/content/m19202/latest/Demo_Standard_Libraries_Listing.txt>

Definition 2: confidence
The reliance that Standard Library functions work properly.

Definition 3: abs
A function within the cmath standard library in C++ which stands for absolute.