UNBC Laboratory Assignment

Computer Science 101—Winter 2007

Pointers and Function Pointers

Due Date:

This assignment is due 02 February 2007 *at the beginning of lecture.*

Purpose:

To learn to use function pointers to provide a general purpose tool.

Producing bar charts

Documentation

There is one question in this assignment. It asks you to produce a function with a certain signature. It should be understood that you are also to write test procedures to test your functions, and properly script your output.

Goals

Your goal in this assignment is to produce a simple plotting tool by writing a function `printBarGraph`, whose signature is given below. Its input is an arbitrary function and an "x" and "y" range. The output should look something like that shown in Figure 1.

In order to be able to supply an arbitrary function to your `printBarGraph` function, you need to be able to pass a pointer to a function that takes a `double` as an argument and returns a `double`. See section 5.11 of Deitel and Deitel.

⇒ Write a routine with signature:

```
// appropriate definition of double_double_function_t
void
printBarGraph(ostream& out,                   // where to print output
              double min_input,               // range for input data
              double max_input,
              int numLines,                   // num lines to use 1 per val
              int maxCols,                    // max total number of columns
                                              // including numbers
              double_double_function_t f) ;   // function to make bar graph of
```

so that a call like

```
-1.60e+00                                (-2.92e-02)
-1.49e+00 ***                            ( 8.11e-02)
-1.38e+00 ******                         ( 1.90e-01)
-1.27e+00 *********                      ( 2.97e-01)
-1.16e+00 ************                   ( 4.01e-01)
-1.05e+00 ***************                ( 4.99e-01)
-9.38e-01 ******************             ( 5.91e-01)
-8.28e-01 ********************           ( 6.77e-01)
-7.17e-01 **********************         ( 7.54e-01)
-6.07e-01 ************************       ( 8.21e-01)
-4.97e-01 **************************     ( 8.79e-01)
-3.86e-01 ***************************    ( 9.26e-01)
-2.76e-01 ****************************   ( 9.62e-01)
-1.66e-01 *****************************  ( 9.86e-01)
-5.52e-02 *****************************  ( 9.98e-01)
 5.52e-02 *****************************  ( 9.98e-01)
 1.66e-01 *****************************  ( 9.86e-01)
 2.76e-01 ****************************   ( 9.62e-01)
 3.86e-01 ***************************    ( 9.26e-01)
 4.97e-01 **************************     ( 8.79e-01)
 6.07e-01 ************************       ( 8.21e-01)
 7.17e-01 **********************         ( 7.54e-01)
 8.28e-01 ********************           ( 6.77e-01)
 9.38e-01 ******************             ( 5.91e-01)
 1.05e+00 ***************                ( 4.99e-01)
 1.16e+00 ************                   ( 4.01e-01)
 1.27e+00 *********                      ( 2.97e-01)
 1.38e+00 ******                         ( 1.90e-01)
 1.49e+00 ***                            ( 8.11e-02)
 1.60e+00                                (-2.92e-02)
```

Figure 1: output from `printBarGraph(std::cout, -1.6, 1.6, 30, 54, &cos) ;`

```
// #include <cmath>
printBarGraph(std::cout, -1.6, 1.6, 30, 54, &cos) ;
```

produces an output somewhat like that shown in Figure 1.

Steps to produce a bar chart

This section describes how to think about the mathematics of figuring out how to compute the number of stars per line. In order to be able to actually perform these calculations, you also need to know how to call a function through a function pointer. Review *Deitel and Deitel* §5.11 .

In order to produce a good bar chart, you should do something like the steps below:

1. Determine the straight-line function that converts a line number (an integer in the range `1` to `numLines`) into an input value (a double in the range `min_input` to `max_input`). It is a straight-line function, so there are numbers m_{in} and b_{in} so that if x is the input value and `L` is the line number then,

   $$x = m_{\text{in}}\texttt{L} + b_{\text{in}}$$

   The values m_{in} and b_{in} depend on the `min_input`, `max_input` and `numLines` values.

If you don't trust your math skills, ask your lab instructor or professor to check how you compute m_{in} and b_{in}.

2. Determine the maximum (most positive) and minimum (most negative) output values that `f` produces over the input range that you are given. Call them $f_{\max}$ and $f_{\min}$.

   In general, determining the maximum and minimum values of a function requires calculus. Here, however, you only want to compute the maximum and minimum values actually produced when creating a the bar graph.

3. Use $f_{\max}$ and $f_{\min}$ to find m_{out} and b_{out} so that the function $g(y) = m_{\text{out}}y + b_{\text{out}}$ satisfies

   $$g(f_{\min}) = 0, \qquad g(f_{\max}) = \texttt{maxCols} - 24$$

   Again, if you don't trust your math skills, ask your lab instructor or professor to check how you compute m_{out} and b_{out}.

4. Once you know m_{in}, b_{in}, m_{out} and b_{out}, you can determine the number of stars to print (`S`) for each line number `L` by using the formulæ

   $$\begin{aligned} x &= m_{\text{in}}\texttt{L} + b_{\text{in}} \\ y &= \texttt{f}(x) \\ c &= m_{\text{out}}y + b_{\text{out}} \\ \texttt{S} &= \lfloor 0.5 + c \rfloor \end{aligned}$$

   Here x, y, and c are `double` quantities, and `L` and `S` are integers.

   You now have enough information to write a simple `for`-loop to plot the function line-by-line.

You may want to consult your `CPSC 100` text or Chapter 12 of *Deitel and Deitel* to determine how to format the numbers on your graph.

Testing for your bar chart program

This section describes the testing that you should perform on your `printBarGraph` function.

⇒ Write your own test routine to test this function, and show the compilation and output for your test routine.

⇒ In order to be sure that your function can handle arbitrary input, try using the functions supplied in the directory `/export/research/casper/cpsc101/lab3/` on `galaxy.unbc.ca`. Plot the function `mountains` on the range $0 \le x \le 1$, and plot the function `hillside` on the range $-1 \le x \le 1$. A `.h` file for the functions can found in `functions.h` in that directory. An object file can be found in `mountains.o`.

Note that you do not get access to the source code for these functions!