UNBC Laboratory Assignment
Computer Science 101—Winter 2007

Strings & Pointers

Purpose:

1. To practice using the `std::string` class. 2. To use pointers to check how memory is organized.

Due Date:

This lab is due at the beginning of class Monday, 22 January.

Strings:

From *Deitel and Deitel*

⇒ **5.33**

Write a program that uses random number generation to create sentences. The program should use four arrays of pointers to `char` called `article`, `noun`, `verb`, and `preposition`. The program should create a sentence by selecting a word at random from each array in the following order: `article`, `noun`, `verb`, `preposition`, `article`, and `noun`. Use the C++ `string` library to create a string to which each word is added as it is picked. (If you are unfamiliar with the C++ `string` library you might want to look at *Deitel and Deitel* Chapter 15.) The words should be separated by spaces. When the final sentence is output, it should start with a capital letter and end with a period. The program should generate 20 such sentences.

The arrays should be filled as follows: the `article` array should contain the articles `"the"`, `"a"`, `"one"`, `"some"`, and `"any"`; the `noun` array should contain the nouns `"boy"`, `"girl"`, `"dog"`, `"town"`, and `"car"`; the `verb` array should contain the verbs `"drove"`, `"jumped"`, `"ran"`, `"walked"`, and `"skipped"`; the `preposition` array should contain the prepositions `"to"`, `"from"`, `"over"`, `"under"`, and `"on"`.

Pointers and memory diagrams:

Pointer review exercise

⇒ Write a routine with signature:
`bool isSorted(double const * beginning, unsigned int length) ;`
that checks to see whether a section of an array is in sorted order. For instance, to test whether the array `data` had entries `[2]`, `[3]`, `[4]`, and `[5]` in order, you would call
`isSorted(&data[2],4)`. Make sure that your routine works correctly when length is very small.

⇒ Write a driver routine that shows that your `isSorted` function works correctly.

Memory diagrams

You can print the hexadecimal location of a variable or object by casting its address to `void*` as in

```
cout << static_cast<void*>(&i) ;
```

Printing the address of function pointers is a little bit more tricky to do in a standard conforming way. The following seems to work:

```
cout << reinterpret_cast<void*>(
          reinterpret_cast<unsigned long>(&main)) ;
```

(This may not work on platforms where the size of an unsigned long is smaller than the size of a function pointer, or where the size of an object pointer is less than the size of a function pointer.)

⇒ Using these ideas, find the relative locations of the heap, stack, global and static memory, and the code regions of a small C++ program.

⇒ We have drawn stack frames as growing "up". Write a short program that determines whether in fact stack frames stack "up" or stack "down".

⇒ Determine as best you can what order local variables are stored inside one stack frame.