Computer Science II — CSci 1200
Lab 9
Dynamic Memory

Introduction

This lab explores dynamic memory allocation. It also continues to explore command-line arguments and file I/O. It will help to refer back to examples from Lab 8.

In addition to working on the mechanics of arrays, you will see the contrast between using vectors and using arrays to solve the same problem.

Checkpoints

1. Write a program that takes a sequence of doubles from the command-line, stores them in a dynamically allocated array, sorts them, and outputs the values in increasing order, one per line. You will have to determine the size of the array from the number of command-line arguments. As an example, running the program may look like:

   ```
   > sortdouble 2.34 -12.2  67.3 88.2 6.7
   -12.2
   2.34
   6.7
   67.3
   88.2
   >
   ```

   You must use the `atof` function to convert each command-line argument (except `argv[0]`) from a C-style string to a double.

   You must use the standard library `sort` function to sort the array. If `a` points to the start of an array, and there are n values that you would like to sort, then the call to `sort` would look like

   ```
   sort( a, a+n );
   ```

   Note that `a+n` points one array location **past** the end of the interval to be sorted.

2. Write a program to read a sequence of strings from a file into a `vector` of strings and then output the strings to `cout` in increasing lexicographic order (the order produced by `operator<` on strings). There are no surprises here; the problem is straightforward and uses techniques you already know.

   Use the source code from your question 1 solution as the input file.

3. Write a program to read a sequence of strings from a file into an `array` of strings, and output the strings to `cout` in increasing lexicographic order. The problem here is that you don't know how big to make the array. Start by dynamicaly allocating an array of some small size, such as 5. Then, when this fills, apply the technique from the Lecture 15 exercise to double the size of the array. This means you need to keep track of the size of the allocated array and the number of strings actually stored in the array. Initially, the number of strings stored is 0 and the allocated array size is 5.