CSC112 Fall 2010
Fundamentals of Computer Science
Lab 5 – Binary files, 1-D Arrays, 2-D Arrays

In this lab we will build upon last week's lab using arrays.

Part I: 1-D Dynamically Allocated Arrays

Create a directory named Lab5. Keep all of your source and compiled programs in the directory Lab5.

Write a C++ program that reverses a binary audio file named "audio.s16". To do this there are several details.

1. You can download the file "audio.s16" from the URL:

   http://menehune.opt.wfu.edu/CSC112/Lab5/audio.s16

2. Use the techniques discussed in class for reading and writing binary files using objects of type **ifstream** and **ofstream**. The file consists of a sequence of 16 bit signed integers. The base type for this data in C++ is "**short int**".

3. Your program will read the entire file and store it in a single array. To allocate this array, you will need to know how many bytes in the file. Use the system function `stat()` to get this information. See the example handout for help using `stat()`. Your array will be an array with base type `short int`. You will need to compute the number of `short int`s that correspond to the file size: divide the number of bytes by the size of a `short int`. Use the built-in C++ function `sizeof(short int)` to get the size of type `short int`.

4. You should create a dynamically allocated array using a pointer variable, `new`, and `delete` to manage storage for the contents of the input file.

5. Your program must check for errors, including:

   (a) Check the return code from the call to `stat()` to ensure that the call succeeded. Read "`man -s 2 stat`" to find out what the return value from `stat()` means.

   (b) When allocating memory using `new`, check that the pointer is not NULL after allocating. This condition indicates that memory allocation failed.

   (c) When opening both the input and output files, check for failure, and issue appropriate error messages in case of failure.

   (d) When reading from a file , and when reading to a file (using "`.read()`" and "`.write()`" member functions of an `ifstream` and an `ofstream` object respectively), check for failure to read the requested number of bytes, and issue an error message if appropriate. Use the "`.fail`" member function to check for failure.

6. Use an adaptation of the function reverse() from last week to reverse the array. Note: the "adaptation part" is because you need to reverse signed shorts instead of unsigned shorts.

7. Write the array to a binary file named "**audio_rev.s16**".

8. Install the **sox** package on your Ubuntu virtual machine.

   (a) Start a privileged (root) session.

   (b) apt-get install sox

   (c) End your privileged (root) session.

9. Use the play command to listen to both the forward and reversed audio. The simple audio format we are using does not store the sample rate in the file, so we have to supply that information on the command line of the `play` command. The sample rate is the standard CD audio rate of 44100 samples per second. To listen to the downloaded audio file, use the command:

   play -r 44100 audio.s16

10. It is time for us to work on programming style.

    (a) Use comments in your program to describe the logic of what your program is doing.

    (b) Indent all control constructs for easier reading.

Part II: 2-D Dynamically Allocated Arrays

In this exercise, we will write a C++ program to flip a picture left-to-right. To this end, there are some details.

1. Download a file named "tux.dat" from:

   http://menehune.opt.wfu.edu/CSC112/Lab5/tux.dat

2. This file is a binary file which represents a black and white image of a friend as a matrix of double precision values (base type double). It has the following format:

   (a) The first 4 bytes are an integer m indicating the number of rows of data.

   (b) The next 4 bytes are an integer n indicating the number of columns of data.

   (c) The next $8 \times m \times n$ bytes represent a matrix of $m \times n$ double precision values (base data type `double`) in **column major order**.

   To read this file, you will need to first read m and n, and then allocate an array to hold the rest of the data from the file.

3. To view this file:

   (a) Download a program named `dat2ppm` from:

   http://menehune.opt.wfu.edu/CSC112/Lab5/dat2ppm

   Make sure the execute permissions are set correctly for `dat2ppm` and put it in `/usr/local/bin`.

   (b) Use the command

   ```
   dat2ppm -gray tux.dat
   ```

   to create a ".ppm" graphics format file.

(c) Use the command "`display tux.ppm`" to view the image.

4. After you have read the data file into a 2-D array, flip the image left-to-right. To do this, loop over all rows of the matrix, and in each row, reverse the row. This is similar in concept to the way you reversed a 1-D array in part I, but be mindful of the storage issues. Adjacent elements in the same row are not stored in adjacent memory locations.

5. After you have flipped the image left-to-write, write it to a data file named **tux_lr.dat**. Use the same data format as described for the input file.

6. You should use `dat2ppm` and `display` to verify that the image has been processed correctly.

Turn in: Change to the directory containing the sub-directory "Lab5" Create a file named "lab5.tar" using the command:

```
tar cf lab5.tar Lab5
```

Upload the file "lab5.tar" to your account on telesto.