Prelab 3 (September 16)

We've discussed how the size/length (number of elements) of an array can be stored before the beginning of an array so that it is accessible whenever needed without having to explicitly pass it as an extra parameter for every function that uses the array. That can eliminate many opportunities for errors when working with multiple arrays. (We'll soon see that eliminating complexities associated with maintaining multiple arrays is the motivation for `structs` in C.) What's most important for this class, however, is how the storing of extra information with arrays provides strong practical experience with the use of pointers, pointer arithmetic, and dynamic memory allocation.

In this prelab you will create functions to permit users to create arrays that can be indexed in ways other than starting from zero. Before C, most programming languages indexed starting at 1 instead of zero. Other languages allow arrays to be declared with specified index limits, e.g., to allow a 200-element array to be indexed from -100 to +100. This can be very convenient when the index range corresponds to something such as employee ID numbers or calendar years.

```
/* This function returns a double array that can be indexed
   starting at minIndex and ending at maxIndex, i.e., minIndex
   is the index of the first element of the array and maxIndex
   is the index of the last element of the array. The function:
   void freeDoubleArray(double * array, int minIndex) must be
   called to free the allocated array.
*/

double * createDoubleArray(int minIndex, int maxIndex);
```

Below is an example of how this function might be used:

```
/* Create an array to that will hold the average global rainfall
   for each year from 1900 to 2000.  */

rainfall = createDoubleArray(1900, 2000);

for (i=1900; i<=2000; i++) fscanf(fp, "%lf", &rainfall[i]);

// Print average rainfall for the year 1962
printf("Avg rainfall in 1962: %lf\n", rainfall[1962]);

// Calculate and print average rainfall during the 1970s
double sum=0.0;
for (i=1970; i<1980; i++) sum += rainfall[i];
printf("Avg rainfall in the 1970s: %lf\n", sum / 10);

freeDoubleArray(rainfall, 1900);
```

Note that the interfaces (prototypes) for the array creation and free functions have been specified, so you have no discretion about that. The question you may have is: *How can I – a mere undergraduate student – perform magic of this kind?* Well, it's a heck of a lot easier than you think. All you need to do is malloc an array of the requested size and then subtract the start index from the array pointer. In other words, the array pointer is no longer really the address of the start of the allocated array. In the example above, it would be 1900 doubles before the real address of the beginning address of the array, so when the calling program/user accesses `rainfall[1900]` they're accessing the real first address location of the array. Think about why a special function is needed to free the array – and why the start index is required as a parameter.

- JKU