COMP220 — Lab 8

Fall 2017

Abstract

For this lab you'll write a simple array comparison procedure that is necessary for testing array equalitys.

Array Testing

Consider the following code:

```
int array1[] = {0, 1, 2};
int array2[] = {0, 1, 2};

EXPECT_EQ(array1 == array2);
```

Although each array has its own block of memory, these arrays are "equal" in the sense that they contain the same elements. However, remember that array variables used without brackets "decay" into pointers! So the test above actually fails, since array1 and array2 decay into different pointers, i.e. they don't have the same starting address.

For a class we would overload the `==` operator for our class. Unfortunately, in C++ you cannot overload operators for built-in types (arrays, pointers, int, char, etc.). So we are left with two options for checking equality of arrays:

1. Manually write a loop that iterates through the expected and actual arrays and compares each individual element. You'll have to write this loop each time you want to compare two array.
2. Write a helper procedure to test if two arrays have the same contents.

The latter option is more reusable and something we'll tackle today.

arrayEquals

Design and implement a procedure named `arrayEquals` that takes two arrays of integers and returns true if they have the same contents and false otherwise. When passing arrays we must pass the array variable and the array size. Thus your procedure has the following signature:

```
bool arrayEquals(int lhs[], int lsize, int rhs[], int rsize);
```

Write a library containing this method, complete with a header, finished implementation, and gTests, as assignment *lab8*.

Due by midnight on Friday, November 3.