CS 185
Programming Assignment 3

Details

This is another short assignment to help you practice with arrays, functions, and iteration. The program will manipulate arrays in several different ways. The elements in all of the arrays are integers. There are five simple functions that you need to write to complete the assignment.

Function	Description

`void`
`reverse_array(int a[], int size);`
	Given an array, reverse the order of the elements in the array. Do **not** create another array in the function.

`void`
`add_arrays(const int a[], const int b[], int c[],`
`           int size);`
	Given three arrays, add the elements of the first two arrays and put the sum in the third array.

`void`
`scalar_multiply(int a[], int size, int multiplier);`
	Given an array and a multiplier, multiply each element by the multiplier.

`int`
`dot_product(const int a[], const int b[], int size);`
	Given two arrays, determine the dot product (multiply each corresponding element and sum the products). Return the value.

`void`
`cross_product(const int a[], const int b[], int c[]);`
	Given three arrays, determine the cross product of the first two. The cross product is another array and will be placed into the third array. The size of all three arrays will always be 3.

Each function provides the size of the array, with the exception of the cross product. (The cross product will always have arrays of size 3.) If there is more than one array, you can be sure that all of them are the same size.

Examples:

`reverse_array:`

Given array **a** with these values: `1 2 3 4 5 6 7 8`

After reversing, **a** looks like this: `8 7 6 5 4 3 2 1`

`add_arrays:`

Given array **a** with these values: `1 2 3 4 5`

Given array **b** with these values: `3 4 7 2 1`

Array **c** will look like this: `4 6 10 6 6`

`scalar_multiply:`

Given array **a** with these values: `1 2 3 4 5`

Given this multiplier: `8`

Array **a** will look like this: `8 16 24 32 40` `dot_product:`

```
    Given array a with these values: 1    2    3
    Given array b with these values: 3    4    7
             The return value is this: 32
```

`cross_product:`

```
    Given array a with these values: 1    2    3
    Given array b with these values: 3    4    7
           Array c will look like this: 2    2   -2
```

Dot and Cross Products (A short review of MAT140)

The **dot product** evaluates to a scalar (number) and is simply the sum of the products all of the corresponding elements of two arrays. So, if we have two arrays of 3 integers each, a and b:

```
int a[] = {1, -4, 5};
int b[] = {3,  1, 2};
```

and we want to calculate the dot product, it would be like this mathematically:

```
a ● b = a1·b1 + a2·b2 + a3·b3
```

Expanding the example:

```
[1  -4   5] * [3   1   2] = (1)(3) + (-4)(1) + (5)(2) = 3 + (-4) + 10 = 9
```

Note that the arrays can be of any size, but both will be the same size. So, if the arrays had 100 elements in each:

```
a ● b = a1·b1 + a2·b2 + a3·b3 + ... + a99·b99
```

The **cross product** evaluates to an array and only works on arrays of size 3. So, if we have two arrays of 3 integers each, a and b, and a third array, c:

```
int a[] = {1, -4, 5};
int b[] = {3,  1, 2};
int c[3];
```

and we want to calculate the cross product, it would be like this mathematically:

```
c = a X b

c1 = a2·b3 - a3·b2
c2 = -(b3·a1 - b1·a3)
c3 = a1·b2 - a2·b1
```

Expanding the example:

```
int a[] = {10, 9, -7};
int b[] = {-2, 4, -5};
int c[3];
```

Calculations:

```
c1 = (9)(-5) - (-7)(4) = -45 - (-28) = -17
c2 = -((-5)(10) - (-2)(-7) = -(-50 - (14)) = -(-64) = 64
c3 = (10)(4) - (9)(-2) = 40 - (-18) = 58
```

So, the array, c, looks like this:

```
[-17, 64, 58]
```

You are given a file called `main.cpp`, which includes the main function. There are five functions prototyped in that file, one for each of the functions you are to write. As always, you must implement all of these functions *exactly as prototyped.* You will implement these functions in a file called `array.cpp`.

An output.sample.txt file is given to you so that you can compare your output with the expected output.

You are also given a file called `main2.cpp`, which contains more tests. Compare the output from main2.cpp with the `output.sample2.txt` file.

Note: ***I expect the exact output. So be careful and use a tool (such as: WinMerge) that will check the difference between two text files.***

This assignment will not require you to include any header files in your code. All of the code that you write must be in ***array.cpp***, since you will not be turning in ***main.cpp*** or ***main2.cpp***.

REPEAT: **You will not turn in main.cpp** or **main2.cpp**, so any changes in them will not be seen by me.

With the exception of the cross product function, all of the functions will be doing some sort of looping. You can use either a **for** loop, **while** loop, or **do** loop. Also, start the assignment by making stub functions. This will get you up and running very quickly. Of course, depending on your stub functions, you may get some compiler warnings, but this is to be expected since you haven't written the real code yet. Also, you are **NOT** to create any arrays in any of the functions you write. All of the arrays that you need are given to you as parameters to the functions. No credit will be given for any function that creates an array.

<u>Note:</u>

- Make sure that your functions (except cross product) can handle any size array (not just the ones that are demonstrated in the main.c files).
- You'll see some #define directives in the second main file. How large can you make the arrays so that the program still works? (Hint: redirect the output to a file, otherwise it will take *forever* to print very large arrays to the screen.)

  When it fails, you'll see something like this (or the program will crash!):

```
  16600  [main]  a  2072  _cygtls::handle_exceptions: Exception: STATUS_STACK_OVERFLOW
  16600  [main]  a  2072  _cygtls::handle_exceptions: Exception: STATUS_STACK_OVERFLOW
  30499  [main]  a  2072  open_stackdumpfile: Dumping stack trace to a.exe.stackdump
  30499  [main]  a  2072  open_stackdumpfile: Dumping stack trace to a.exe.stackdump
 788004  [main]  a  2072  _cygtls::handle_exceptions: Exception: STATUS_ACCESS_VIOLATION
 803100  [main]  a  2072  _cygtls::handle_exceptions: Error while dumping state (probably
                          corrupted stack)
```

  What do you think the problem is?

Comments

In this and future assignments, you are required to include:

- **A file header comment in every piece of source file. The format is shown in the "Comments.cpp" file given to you in the beginning of the semester and should be present at the very top of all your code.**
- **Function header for each function you create. The format is shown in the "Comments.cpp" file given to you in the beginning of the semester and should be present at the top of every function.**
- **Inline commenting for your code.**

What to submit

You must submit the CPP file (***array.cpp***) in a single .zip file (go to the class page on moodle and you will find the assignment submit link). ***Do not submit any other files than the ones listed.***

If you've forgotten how to submit files, the details are posted in the syllabus and in the assignment guidelines document. Failure to follow the instructions will result in a poor score on the assignment (and possibly a zero).

Special note:

The due date/time posted is the positively latest you are allowed to submit your code. Since the assignments can easily be completed well before the deadline, you should strive to turn it in as early as possible. If you wait until the deadline, and you encounter unforeseen circumstances (like being sick, or your car breaking down, or something else), you may not have any way to submit the assignment on time. Moral: **Don't wait until the last day to do your homework.**