CS1580 Lab 8

**Background**

Arrays

Arrays are a contiguous list of a type of data. Each value right next to the last. This allows us to index into the array to get the value we need. You can declare an array:

<type> <name>[<const\_expr>];

<type> <name>[<const\_expr>] = <initializer\_list>;

<const\_expr> is optional in the second case.

<initializer\_list> = {<expr\_list>}

<expr\_list> = <expr> [, <expr\_list>]

<const\_expr> is a value that can be determined at compile time.

There are come functions that qualify for const\_expr but, for the purposes of this class it will be constants or literals (which would be a bad magic number so, just constants).

Arrays start at index 0. Think of index as an offset into the array. This makes the first element an offset of 0. This is because of how the compiler calculates the memory address: A + type\_size\*index.

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| Value | 5 | 1 | 7 | 3 | 9 | 5 | 1 | 7 | 0 | 8 |

You can reference a value of a array by the syntax:

<name>[<expr>]

Where the <expr> here should resolve as an index into the array.

and to assign a value:

<name>[<expr>] = <expr>;

Where the first <expr> should resolve as an index into the array and second should be the value to assign to that location.

***Note*** that referencing just the array will not get the whole array. It will get the pointer for the array which rest of the array can be accessed from. That being said the following will not print the array. It will instead print the pointer for the array.

int a[4] = {0,1,2,3};

cout << a << endl; // 0x7fff...

***Careful*** though, C++ doesn’t check if the index is valid. If the index is negative or greater than or equal to the size of the array the value being accessed will ***not*** be in the array. This is called “walking off the array” and it ***will*** compile, though you ***may*** get a segmentation fault (basically your program trespassed and the OS said “Hell, no”).

Try the following and see what happens:

#include <iostream>

using namespace std;

const int SIZE = 8;

int main()

{

int a[SIZE] = {0,1,2,3,4,5,6,7};

int b[SIZE] = {8,9,10,11,12,13,14,15};

std::cout << “a: “ << a << std::endl << “b: “ << b << std::endl;

std::cout << “a[0]: ” << a[0] << “ a[7]: “ << a[7] << std::endl;

std::cout << “b[0]: “ << b[0] << “ b[7]: “ << b[7] << std::endl;

// clang? Swap a and b these two lines

std::cout << “b[-1]: << b[-1] << std::endl;

std::cout << “a[8]: << a[SIZE] << std::endl;

return 0;

}

You can pass arrays to functions as well. Their type is <type>[].

void array\_fn(int[] array, const int size);

Notice how this function also takes the size of the array. This is so the function can avoid walking of the array itself.

**Example:**

const int A\_SIZE = 7;

const int B\_SIZE = 8;

const int C\_SIZE = 4;

void array\_fn(int[] array, const int size)

{

for(int i = 0; i < size; i++) {

array[i] = rand() % 100;

}

return;

}

int main()

{

int a[] = {0,1,2,3,4,5,6};

int b[B\_SIZE];

int c[C\_SIZE] = {0,1};

cout << a[4] << endl; // 4

a[4] = 8;

cout << a[4] << endl; // 8

array\_fn(b, B\_SIZE); // assigns values to b

cout << c[0] << “ “ << c[1] << endl; // 0 1

cout << c[2] << “ “ << c[3] << endl; // 0 0

return 0;

}

Bubble Sort

[Sorts](https://www.youtube.com/watch?v=ZZuD6iUe3Pc)

Bubble Sort is the simplest of sorting algorithms. For an array n elements long, it goes through the array n-1 times and swaps adjacent elements that are out of order.

Take the array:

First we would take the first two elements and compare them (3 < 6), they are in order so, we move up

compare 3 > 6

and do the same for the next two elements (6 > 2), they are out of order so, we swap them.

compare 6 > 2

swap

Then, we continue until we get to the end swapping pairs of elements if they are out of order.

compare 6 > 9

compare 9 > 12

compare 12 > 1

swap

compare 12 > 8

swap

compare 12 > 4

swap

Notice now that 12 is now sorted at the end of the list. This will always be true so successive passes we won’t have to traverse the entire array.

compare 3 > 2

swap

compare 3 > 6

compare 6 > 9

compare 9 > 1

swap

compare 9 > 8

swap

compare 9 > 4

swap

Now the second pass is done and now 9 is sorted so, again we can sort one less next time.

The next passes will result in (**bolds** are sorted)

Here is a video going over bubble sort: [Bubble Sort | GeeksforGeeks](https://www.youtube.com/watch?v=nmhjrI-aW5o)

Lab Topics

Arrays

Sorting

Task

This lab I’m giving you a main file which you will use. In this main file you’ll update the include for your files. It has four functions inside it:

void generate\_array(int array[], const int size);

This will take user input to fill the array. No printing.

void print\_array(int array[], const int size);

This will print the array with the format “[ 1, 2, 3, … ]”

void sort\_array(int array[], const int size);

This will sort the array from smallest to largest. You can use any sort method you wish though I recommend you use Bubble Sort as we went over this in class. No printing.

int find(int array[], const int size, const int value);

Make a function to find the index of a value (-1 if it doesn’t exist)

This is what is required for the class and what will be graded. I encourage you to experiment with your code. Before continuing submit your assignment and run the command git checkout -b extra. This will make a new branch in your code to separate your submission code and your play code. Here are some suggestions:

Make a function to generate a random array.

Make template functions of the above functions.

Make a function to find the median of the array.

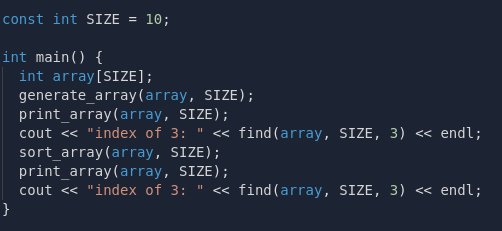
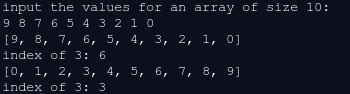
Make a function to find the mean of the array.

Make a function to find the standard deviation.

This is the square root of the sum of the squares of the differences of each value and the mean then divided by the number of elements.

;

Sample input and output

How to get full points

Refer to the rubrik:  
 Section 301: https://mst.instructure.com/courses/53135/assignments/syllabus

Section 303: https://mst.instructure.com/courses/50476/assignments/syllabus

I expect to see at least one while/do..while and at least one processing arrays.

Your main.cpp file will be given to you. You may change it as you wish, as it will not be graded.

I will use my own main to test you functions individually. This means your functions ***must*** match the above prototypes ***exactly***.

Lab Preparation

10am <section> = 301, 6pm <section> = 303, <username> = your university username

if ssh:

git clone [git@github.com](mailto:git@github.com):mst\_sp21\_cs\_1580\_<section>/2021-sp-<section>-lab8-<username>

if https:

git clone <https://git-classes.mst.edu/mst_sp21_cs_1580>\_<section>/2021-sp-<section>-lab8-<username>

cd 2021-sp-<section>-lab8-<username>

How to submit

Git:

git add .

git commit -m “your message”

git push

Canvas:

Add each cpp,h,hpp,tpp file to submit.