# Project 2: 2D arrays

# CS111 ProGramming II

## Purpose

Create a project that uses 2D array to simulate temperature profile in a container of water.

## Due Date

The project is due as specified on Schoology. Late assignments will not be accepted.

## Deliverables

There are two deliverables for this project:

* Turn in the complete project. The project must be named CS111Project2-Lastname, where Lastname is your last name. If you do not follow this format, you will receive a zero for the project.
* Completed grading rubric with scores and justification.

Note: The entire Visual Studio or Xcode project must be turned in. If you turn in only the .cpp file (the C++ file) or your project does not build, you will receive a 0 (zero) on the project.

If you have any questions, please see me well before the due date.

## Grading and Student Learning Outcomes

**Total points for this assignment: 50**

This project applies to Student Learning Outcomes 2 and 6.

The project is graded using the rubric (additional file).

**This is an individual assignment. You may not show your code to anyone except the professor and tutors. You may not look at anyone else’s code under any circumstances.**

## Code Specification

Recall in lecture we worked through a 2D array to calculate a temperature profile. In this assignment, you will create a version of that project.

Create a new project that has the following items:

* typedef for a 2D array of double called *Matrix.* The size is 15.
* An instance of matrix called *simulation* in main()
* Due to various losses, **each cell gets only 98% of the value of the neighboring cells**. Thus, you must modify the function we used in class.
* The following functions:

//precondition: matrix is the simulation, size is number of columns and rows

//postcondition: matrix is initialized to border temperature.

// Ask the user to supply a boundary temperature greater than 0

// All non-border elements are all set to zero.

void initializeSimulation(Matrix matrix, int size);

//precondition: matrix is the simulation, size is number of columns and rows,

// runs is the of simulation runs.

//postcondition: matrix contains the simulation values after the simulation completes

// *runs* runs.

void simulate(Matrix matrix, int size, int runs)

//precondition: matrix is the simulation, size is number of columns and rows

//postcondition: the contents of the matrix are output to the screen

// set precision to one and width to four

void printSimulationResults(Matrix matrix, int size, int runs)

In *main(),* you will need to:

* declare necessary variables,
* properly call the functions,
* use the average function we used in class, and
* run the simulation **20** times.

You will need additional functions; ensure they are properly commented.

Be sure to properly average the cells in the simulation; you can’t run off the end of the array.

Extra credit (10 points)

Instead of running the simulation for *20* runs, run the simulation until no cell changes more than a tolerance given by the user. Tolerances for this type of simulation are usually between 0.001% and 0.1%.