# Week 5

## Announcements

- Project 4
    - 11/12 (next Thursday!) 11:00 PM
    - zybook exercises

## Questions?

- anything?

## Arrays

```
// un-initialized
int scores[10];     // garbage values
int s1 = scores[0]; // don't know what s1 holds

// initialized
int scores[10] = {1, 2, 3, 4, 5, ..., 10};
/*
you can alse partially initialize e.g. we can do
int scores[10] = {1, 2, 3, 4, 5};
correction from what I said in discussion - the rest of the values must be set to 0
so scores = {1, 2, 3, 4, 5, 0, 0, 0, 0, 0}
*/

scores[2];  // has the value 3
scores[10]; // 9 is the last index, this is a run time error but will compile

int scores2[3] = {10, 12, 99};
/*
assuming each entry in the array is 4 bytes, memory could look like this:

Memory Address: | 1096 | 1100 | 1104 | 1108 | 1110 | 1114 |
Data:           |      |  10  |  12  |  99  |      |      |

if you access indices < 0 or >= 3, you access memory who's contents you don't
know about
*/
```

## Array Argument to Functions

```
// store names of everyone in our discussion with an array

const int NUMBER_STUDENTS = 20;    // good practice to use var for array size
string names[NUMBER_STUDENTS];

for (int i = 0; i < NUMBER_STUDENTS; i++) {
  string currentName;   // you can also directly pass names[i] to getline

  cout << "Enter name: ";
  getline(cin, currentName);

  names[i] = currentName;
}

// pass it to a function
// we're not changing the values of the array, so we make it const
void printNames(const string names[], int numberStudents) {

  for (int i = 0; i < numberStudents; i++) {
    cout << names[i] << endl;
    names[i] = "hi";    // compilation error because names is const
  }
}

// small example
// take strings as input, store them in an array
// cout each string reversed and return the average length
/*
names = ["taasin", "michael"], numberStudents = 2
output:
"nisaat"
"leahcim"
return (6 + 7)/2 = 6.5
*/

double backwards(const string names[], int numberStudents) {
  // if you want to access the names in reverse order, the for loop looks like:
  // for (int i = numberStudents - 1; i >= 0; i--) {

  int sum = 0;

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

    sum += names[i].size();

    for (int j = names[i].size() - 1; j >= 0 ; j--)
      cout << names[i].at(j);

    cout << endl;
  }

  return (static_cast<double>(sum)/numberStudents);
  // you can cast either sum or numberStudents,
  // but if you cast static_cast<double>(sum/numberStudents) you'll still do
  // integer division and then cast it to a double, which isn't what we want
}
```

## 2D Arrays

```
// initialize
int array[6][4];
array[0][1];     // first row, second column (don't forget we count from 0)
                 // value is 9

   0   1   2   3   4
0 [0] [9] [0] [0] [0]
1 [0] [0] [0] [0] [0]
2 [0] [0] [0] [0] [0]
3 [0] [0] [0] [0] [0]
4 [0] [0] [0] [0] [0]
5 [0] [0] [0] [0] [0]
6 [0] [0] [0] [0] [0]


// lets initialize an array that represents a computer screen.
// each element of the array is a pixel,
// and lets say the screen is 50 rows and 100 cols
// 0 represents white, so make sure we start with a white screen

const int ROWS    = 50;
const int COLUMNS = 100;
int screen[ROWS][COLUMNS];

for (int i = 0; i < ROWS; i++) {
  for (int j = 0; j < COLUMNS; j++) {
    screen[i][j] = 0;
  }
}
// this code accesses each row, then each column
// to access each column, then row:
    // switch ROWS with COLUMNS and screen[i][j] with screen[j][i]

// now lets draw a square, aka set all values adjacent to the center to 1
// result:
   0   1   2   3   4
0 [0] [0] [0] [0] [0]
1 [0] [0] [0] [0] [0]
2 [0] [1] [1] [1] [0]
```

```
3 [0] [1] [0] [1] [0]
4 [0] [1] [1] [1] [0]
5 [0] [0] [0] [0] [0]
6 [0] [0] [0] [0] [0]

const int ROWS    = 7;
const int COLUMNS = 5;

int midRow = ROWS/2;
int midCol = COLUMNS/2;

for (int i = midRow - 1; i < midRow + 2; i++) {
  for (int j = midCol - 1; j < midCol + 2; j++) {
    if ( !(i != midRow && j != midCol) )  // think about a different way to do this
      screen[i][j] = 1;
  }
}
```