EECS 183 Discussion

ARRAYS

Arrays

- An array is a collection of items of the same data type that are stored sequentially in memory
- Arrays are indexed starting at 0
- When passed into a function, they are ALWAYS passed by reference
  - This is the default for arrays: you don't need an & to pass an array by reference
  - Arrays are the only thing in c++ for which reference is the default. You will learn more when we cover pointers

Arrays cont:

- You can think of an array like it is an apartment complex:
  - The whole apartment complex has a name, but each unit has a numeric address

```
int highlandApts[3] = {22, 33, 12};
```

highlandApts

Index	0 – highlandAPts[0]	1	2
Contents	22	33	12

Array Declaration Syntax

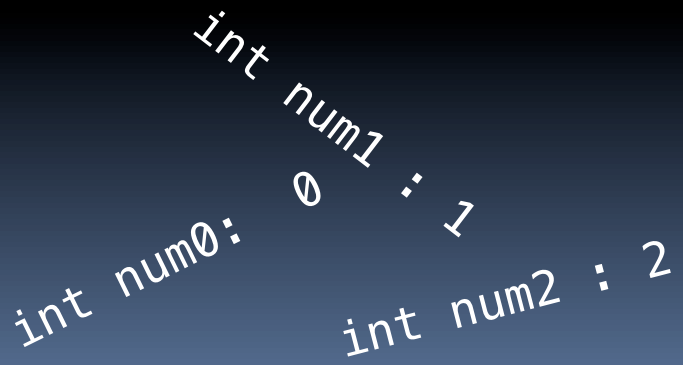
```
int arr[5];
char letters[3] = {‘a’, ‘b’, ‘c’};

const int MAX_SIZE = 4;//need the const
int array[MAZ_SIZE] = {4};
```

//Array size MUST be known at compile time

Arrays in Memory

```
int num0 = 0;
int num1 = 1;
int num2 = 2;
```

IN MEMORY:




```
int num[3] = {0, 1, 2};
```

IN MEMORY:

```
int num[0] : 0
int num[1] : 1
int num[2] : 2
```

Functions and arrays!!

```
const int SIZE = 4;

int main() {
   int nums[] = {1, 2, 3, 2};
   int size = 3;
   multBy3(nums, size);
   printArray(nums, 3);
}
```

//What does this print?

3, 6, 9

```
void multBy3(int arr[SIZE], int
   size) {
   int i = 0;
   for ( ; i < size; ++i) {
       arr[i] *= 3;
   }
}

 void printArray(int arr[SIZE],
   int size) {
   int i = 0;
   for ( ; i < size, ++i) {
       cout << arr[i] << “, “;
   }
}
```

Strings as Arrays

By definition, a string is just an array of characters. Thus, they can be used exactly like arrays in many ways

```
string cool = “EECS ROCKS!”;
cout << cool[0]; //prints E


//What does this print?
int end = cool.length();
for (int i = 0; i < end; ++i) {

    cout << cool[i];

}
```

Pass by Const Reference

- `int foo(const int &a, const double arr[SIZE]);`

- When to use const ref?
  - If you don't want to modify the array being passed to a function.

Does this compile?

```
int bar(const int
  ar[SIZE]) {
    ar[14] = 4;
    return ar[14];
}
```

```
int main() {
  int ar[] = {24};

  cout << bar(ar);
}
```

Multidimensional Arrays

- A 2D array is basically a grid
- Syntax:

```
const int ROWS = 3;
const int COLUMS = 4;
int multi[ROWS][COLUMNS] = {0};

//What does multi now contain?
for (int i = 0; i < ROWS; ++i) {
  for (int s = 0; s < COLS, ++s) {
      multi[i][s] = i + s;
  }
}
```