C++ Crash Course

Module 4: Arrays and Functions

Human Computer Interaction
IOWA STATE UNIVERSITY
OF SCIENCE AND TECHNOLOGY

Arrays and Functions

- Numeric Arrays
- Multidimensional Arrays
- Character Arrays
- Strings
- Array Sorting
- Functions
- Functions with Arrays

Arrays

- Imagine you were writing code to record a vehicle's velocity every second for an hour. How would you store that information?
  - double velocity1, velocity2, velocity3, …, velocity3600
- What if you want to average all those values?
- Arrays provide a better way to deal with situations like this.

Numeric Arrays

- Declaring and initializing arrays
  - *Element type* *name*[*number of elements*];
  - int simpleArray[5];
    - Creates space for five integers
    - {0, 0, 0, 0, 0}

simpleArray[0] simpleArray[4]

  - You can initialize by element or in the declaration.
    - simpleArray[0]=1; → {1, 0, 0, 0, 0}
    - int simpleArray[5] = {1, 2, 3}; → {1, 2, 3, 0, 0}

Numeric Arrays

- Other usage stuff
  - ```
    double array2[40];
    double value = 0;
    for(int i=0; i<40; i++){
      array2[ i ] = value;
      value += 0.25;
    }
    ```
    → {0, 0.25, 0.50, …, 9.75}
  - ```
    value = array2[1] + array2[39];
    ```
    → value = 10.0;

Multidimensional Arrays

- double elevation[10][10];

j \ i	0	1	2	3	4	5	6	7	8	9
0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
9	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

- double rainfall[year][month][day];

Character Arrays

- char is only designed to hold a single character, but we often want words, sentences, etc…
- You've already seen a character array in the keyboard input example code.
- char name[15];

  name = "Andrew";

  char name[ ] = "Andrew";
- Character arrays all end with the '\0' null character.

- chararrays.cpp for examples.

Strings

- Strings are a special class which are similar character arrays, just easier to use.
- Requires the <string> include.
- string firstName = “Andrew”;
- <string> has many useful functions that will work on both strings and character arrays.
- String objects also have methods.
  - firstName.erase(3); ➡ “And”

- Sorting.cpp has some string examples

Arrays

- Keep track of array limits.
  - Out-of-bounds error
- Always remember that indexing starts at zero.
- Array Limitations
  - Fixed size
  - Elements have a fixed index or location
- Alternatives
  - Array classes, Linked lists, STL Maps and vectors

Array Sorting

- Arrays aren't always entered in a way that is useful.
- You'll want to rearrange it.
  - Increasing/decreasing value
  - Alphabetic
  - ??
- When sorting arrays, you must swap values.
  - Need a temp variable to store one of the values.

- Sorting.cpp has an example and an exercise for you to complete.

Functions

- You already have some experience with functions.
  - int main() { }
  - pow(x, 2)

- Functions operate as modular pieces of code. Instead of writing many similar commands, we can create a function to simplify the program.

Functions

- *return type* *name*(*arguments*)
  { //code }

- The return type is what kind of value the function sends back to where it was called, if any.

- The name is whatever you choose to name the function. Again, existing keywords are off limits.

- Arguments are the values you supply to the function when calling it, if any.

Functions

- No return type, no arguments
  - void resetGame(void)

```
{
  score = 0;
  numLives = 3;
  health = 100;
}

int main()
{
  resetGame();
}
```

Must use void keyword to indicate no return type.

Leaving the parentheses empty is also valid and commonly seen:

void resetGame() { }

Calling a no return type function. Do not put void in these parentheses.

Functions

- Single value return type, multiple arguments
  - int findMax(int a, int b)

```
{
  if(a > b)
    return a;
  else
    return b;
}

int main()
{
  int result = findMax(10, 5);
}
```

Declare the return type as int. The function now must return an integer.

Declare the argument variables. These store whatever you call the function with, and you can use them as variables only within this function.

The return keyword is what sends back the int variable.

This function call stores the return value in the variable 'result'. The 10 and 5 values could easily be replaced by variables.

Functions

- As with variables, a function needs to be declared before it can be used.
  - You can write the whole function above the line where you call it…
  - Or you can include a function prototype at the top of the file. The function can then be used anywhere in the file.

- int findMax(int, int);

- Functions.cpp for example code

Functions with Arrays

- You can directly use arrays as arguments to a function.
  - void someFunction(int someArray[ ]) { }
  - Then call it with:

    int numArray[5] = {1, 2, 3, 4, 5};

    someFunction(numArray);
  - This is a “pass by reference” operation. Any changes made to the array within the function will affect the original array.
  - Keep in mind that you may also want to know the size of the array within the function. See the sample code for an example.

Functions with Arrays

- However, you cannot use an array as a return type.
  - ~~double[ ] someFunction() { }~~

- Array outputs from a function is possible with something called a pointer.
  - Example in arrays.cpp

Functions with Arrays

```
// function that returns an array
int* functionArrayOutput(int arraySize) {
    int *arrayToReturn;

    // we allocate the memory
    arrayToReturn = new int [arraySize];

    for (int i=0; i<arraySize; i++) {
        arrayToReturn[i] = i;
    }

    return arrayToReturn;
}
```