

## Computer Science 3-4

### Problem Set #8: Intro to Arrays

An **array** is a powerful *data structure* in C++ that lets us keep track of lists of different types of information. They are quite flexible in that we can put almost anything we want into an array and then access it at a later time. Arrays have two main requirements when we define them: first of all we must give them a type...int, float, string, char, etc. Second of all we need to specify the length of the array. Here are a few examples:

```
int scores[10]; // this creates an array that can hold 10 integers
char word[5]; // this creates an array that can hold a 5 characters (5 letter word)
float list[60]; // this creates an array that can hold 60 floats.

int evens[4] = {2,4,6,8}; // Here we've defined our array size to be 4 and initialized our
    array to hold these 4 elements.

char letters[3] = {'a','b','c'}; // We can initialize a char array the same way;

//If we want to change the value of an item in an array, we select by the index number
// and assign in the normal way.

evens[0] = 10; // This would change the first item in the array above from a 2 to a 10.
letters[2] = 'd' // This would change the 'c' in the above example to a 'd'
```

We call the number inside the brackets the *index* of the array.

#### Some Important Considerations:

1. The first item in the array is always stored at index 0. This is **very, very, very important**. So the 2<sup>nd</sup> item in the array is actually at index 1, the 3<sup>rd</sup> at index 2, etc.
2. You **cannot** directly copy an array, even if they are the same size for example you **cannot** write:

Evens[] = otherEvens[]

To accomplish this you actually need to use a *for* loop to carefully copy each element over one by one. Here's an example of how we can set all of the elements of an array equal to 0 using a for loop:

```
for( int i = 0; i<10; i++)
    n[i] = 0; // We'll put a 0 in at each place of the array as we count up
```

This is the fundamental process we are going to be working with when we work with arrays....we will have some sort of *for* loop that we will use to access or fill up the array. This loop prints out everything in an array:

```
for( int i = 0; i<10; i++)
    cout << "Element "<< i << " Has Value: " << n[i];
```

Here is the standard input loop for arrays:

```
for( int i = 0; i<10; i++)
{
    cout << "Please Enter a Number:  ";
    cin >> n[i];
}
```

We can do all of the same arithmetic we can do with normal numbers using arrays (provided the information stored in the array is actually a double or int). Remember that we start indexing from 0! For instance, if we had the two following arrays:

```
int odd[5] = {1,3,5,7,9};
int even[5] = {2,4,6,8,10};
```

Then:

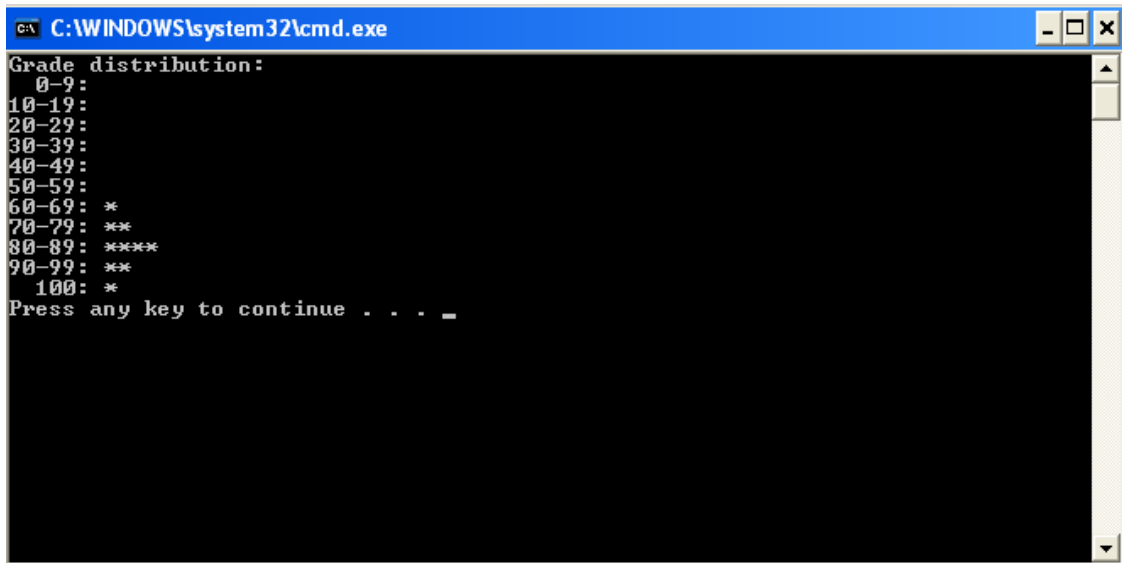
```
sum = odd[1] + even[3];    // This is 3 + 8 = 11
product = odd[2] * even[0]; // This is 5 * 2 = 10
odd[4] = odd[0];          // This would turn the 9 into a 1 in the above list {1,3,5,7,1}
number = even[1+2];       // This would assign the value 8 to number (i.e. even[3])
```

We will be doing a lot with arrays moving forward...here are a few problems to help you get comfortable!

1. **Grades Revisited** – Ask the user how many grades they'd like to enter, store them in an array, calculate the mean and then print out a list of the grades that are below the mean.
2. **Roll the Dice** – We often want to keep track of the frequencies of something occurring. If we set up our array properly, we can use it to keep track of quite a few things we might use counters for in the past (think of it as an array of counters almost...you don't care about the order in which the rolls occur). For this problem, write a program that asks the user how many times they would like to roll a 6-sided die and then keep track of how many of each roll occurred. Print the results to the screen.
3. **Queen's Choice** – Suppose that we want to place eight queens on a standard 8 x 8 chessboard so that we have exactly one queen in each row. Write a program that lets the user input which column the queen is located in each row and then prints out a visual of the board. For example if the user enters: 6,3,4,0,2,7,6,7 we get:

```
*****Q*
***Q*****
****Q***
Q*****
**Q*****
*****Q
*****Q*
*****Q
```

4. **Histogram** – Modify Problem #1 so that after the user has entered in their numbers, the program prints out a histogram (bar graph) of how many values fell between a certain range (i.e. 0 – 9, 10 – 19, etc.). Here is an example of what it should look like (Hint: you can be clever with % here and avoid a bunch of if statements):



```
C:\WINDOWS\system32\cmd.exe
Grade distribution:
 0-9:
10-19:
20-29:
30-39:
40-49:
50-59:
60-69: *
70-79: **
80-89: ****
90-99: **
100: *
Press any key to continue . . . _
```

5. **Making a List...Checking it Twice** – Write a program that generates 20 random integers between 1 and 20 and:
- Prints it to the screen
  - Prints the same list to the screen, but skips any number that has already been printed to the screen (i.e. no number should appear more than once).
6. **A Unique List** – Your task is to fill up an array with 20 random integers (from 1 to 100), however there can be no repeats within the 20 random numbers. **Note:** this is slightly different than the above problem where you have 20 numbers but might only print out 12 of them!