Arrays and Functions

Lecture 20
COP 3014 Spring 2017

February 21, 2017

Storing Arrays

Things to note about C-style arrays:

- An array is not a type
- An array is a primitive C-style construct that consists of many items stored consecutively and accessed through a single variable name (and indexing)
- This is actually done by remembering the starting address of an array, and computing an offset
- The name of an array acts as a special kind of variable – a **pointer** – which stores the starting address of the array

Arrays as Parameters

An array can be passed into a function as a parameter

- Because an array is not a single item, the array contents are not passed "by value" as we are used to with normal variables
  - The normal meaning of "pass by value" is that the actual argument value is copied into a local formal parameter variable
  - In the case of arrays, just the pointer is copied as a parameter. We'll see this in more detail when we get to pointers
- When an array is sent into a function, only its starting address is really sent
- This means the function will always have access to the actual array sent in
- Returning an array from a function works similarly, but we need pointers to use them well (not yet covered)

Examples

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

Note that:

- The varibale arr acts as the local array name for the function
- There is no number in the brackets. int [] indicates that this is an array parameter, for an array of type int
- It's a good idea to pass in the array size as well, as another parameter. This helps make a function work for any size array

Sample call to the above function:

```
int list[5] = {2, 4, 6, 8, 10};
PrintArray(list, 5); // will print:  2 4 6 8 10
```

Using const with array parameters

- When passing an array into a function, the function will have access to the contents of the original array!
- Some functions that should change the original array.
- What if there are functions that should not alter the array contents?
- Put const in front of the array parameter to guarantee that the array contents will not be changed by the function:

```
void PrintArray (const int arr[], const int size)
{
    for (int i = 0; i <size; i++)
        cout <<arr[i] <<' ';
}
```