Lecture 11

Chapter 5.1, 5.2
Topics

• Declaring and referencing an array
• For loops with arrays
• Defined constants for the size.
• Arrays in memory
• Initializing arrays
• Indexing variables as function arguments
• Array parameters
  – Constant array parameter
• Returning an array
Declaring and referencing an array

• To declare an array give it a base type, a name, and a number of elements (declared size)
  – int score[5];
• This is like saying
  – int score0, score1, score2, score3, score4;
• To access one of the five variables in the array
  – score[0], score[1], score[2], score[3], score[4];
  – Note: it always starts at zero (in C++)
For loops with arrays

• Loops and arrays go well together
• Instead of
  
  if(temp1 < 0)
  
    temp1 *= -1;
  
  if(temp2 < 0)
  
    temp2 *= -1;
  
  ...

For loops with arrays

• Loops and arrays go well together
• It could be written this way

```c
for( int i = 0; i < 5; i++ ){
    if( temp[ i ] < 0 )
        temp[ i ] *= -1;
}
```

• What happens if i = 5 and then array was declared with 5 values?
Defined constants for the size.

cin >> number;
int score[number]; //This is illegal use

• int score[5];
  – This use is okay but then you have to remember the number of students.

• const int NUM_STUDENTS = 5;
• int score[NUM_STUDENTS];
  – This is the best usage
    • Don’t have to remember the number
    • If the number changes, it only needs to be modified in one location.
Arrays in memory

• When int x[3] is declare, the compiler will reserve 3 ints worth of memory
• So if x[0] located at memory location 1028 and ints take 2 bytes.

<table>
<thead>
<tr>
<th>Memory Location</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1028</td>
<td>x[0]</td>
</tr>
<tr>
<td>1029</td>
<td>x[0]</td>
</tr>
<tr>
<td>1030</td>
<td>x[1]</td>
</tr>
<tr>
<td>1031</td>
<td>x[1]</td>
</tr>
<tr>
<td>1032</td>
<td>x[2]</td>
</tr>
<tr>
<td>1033</td>
<td>x[2]</td>
</tr>
</tbody>
</table>
Initializing arrays

```c
int score1 = 100, score2 = 90, score3 = 80;
    Is equivalent to
int score[3] = {100, 90, 80};
    Or
int score[3];
score[0] = 100;
score[1] = 90;
score[2] = 80;
    One can even write it this way
int score[] = {100, 90, 80};
```
Indexing variables as function arguments

• Given that
  – double score[] = {1, .90, .80};
  – double n = .90;
• function(n);
  – Is equivalent to
• function(score[1]);
Array parameters

The following is a legal function call.

```java
void average(int a[], int size); //Prototype
...
int score[5] = {90, 75, 80, 100, 80};
int numberOfScores = 5;
...
average(score, numberOfScores);
```
Constant Array Parameter

To make sure that the previous code doesn’t change your scores the following can be done

```c
... void average(const int a[], int size); //Prototype
... void average(const int a[], int size) //Definition
```

This ensures that the function will not change your values.

- Useful if you pass by reference or want to be sure the values are not changed.
Returning an array

• With your current knowledge it is not possible.
• But the way to do it is to return a pointer to the first value in the array.