Arrays in Functions

1E3
Topic 12

Objectives

- This topic should allow students to
  - Pass array elements to functions
  - Pass whole arrays to functions
  - Write functions that update arrays

- This topic is based on sections of Chapter ?? of the textbook.
Functions

- Why do we use functions?
- Why would be want to pass an array element to a function?
- Why would be want to pass an array to a function?
- Revisit the Functions topic if necessary.

Passing in array elements

- If `even(n)` is a function that takes an integer
- And `nums` is an array of 10 integers then the following are valid
  - `even(nums[3])`;
  - `even(nums[i])`;  // if i is e.g. 4
Exercise

- Write a program to square a sequence of 10 numbers.
- E.g. user enters 1, 2.5, 4, 3.2, …
- Prints out e.g. 1, 6.25, 16, 10.24, …
- Use an array to hold the numbers.
- Use a function square(x).

Solutions

- squaresequence.cpp squares each number as it prints them out
  - `cout << square(numbers[i]);`
- squaresequenceA.cpp replaces the elements of the original array by the squared values
  - `numbers[i] = square(numbers[i]);`
- This version leads on nicely to the next topic.
Functions that take arrays

- You can pass a whole array to a function.
- The problem is the function needs to be told the size of the array.
- Suppose sum_array adds up the numbers in an array
  - \( x = \text{sum\_array}(\text{numbers}, \text{ARRAY\_SIZE}); \)
  - The first parameter tells the function where to find the array.
  - The second one tells it how big it is.

Definition of sum_array

- Declaration:
- \( \text{double sum\_array} \)
  - (double \( a[] \), int size);
- Inside the definition you access elements of array \( a \) as \( a[i] \)
  - \( \text{E.g. total} = \text{total} + a[i]; \)
- See sumsequence.cpp
Array parameters are modifiable

- Functions that take arrays can modify the elements of the array.
  - Thus array parameters are like reference parameters.
  - Without a need for the &.
- Use this fact to turn squaresequenceA.cpp into a version that uses functions.
  - See squaresequenceB.cpp

square_array function

- void square_array (double a[], size)
  
  {... a[i] = square (a[i]); ...}
- The line shown alters the value of the ith slot of whatever array was passed into the function as a.
- Calling the function:
  
  square_array(numbers, ARRAY_SIZE);
- The values in numbers are changed by the function square_array.
Reading into arrays

- Write a function to read a list of grades or words into an array.
  - E.g. for firstarray program or reverse sequence
- Again because array parameters are modifiable (without &) this is easy.
- The reading function needs to know how many elements to read.

Read_array function

```c
void read_array(double a[], int asize) {
    //reads asize values into array a
    int i;
    for (i = 0; i < asize; i++)
        cin >> a[i];
}
```

- See readarray.cpp which is a version of firstarray.cpp that uses the above function


**Const modifier**

- By default functions can modify their array parameters.
- If they don’t modify the array, it is good practice to
  - indicate that to the users and
  - make the compiler prevent inadvertent modification.
- \[
  \text{double } \text{sum}\_\text{array} \\
  \quad (\text{const double } a[], \text{int } \text{size});
\]
- \text{Const} says that \text{sum}\_\text{array} function does not alter elements of the array parameter.

---

**Remember**

- Functions that process arrays should take a size parameter.
- Functions that do not modify the array parameter should use the const modifier.
Which function reads in a?

A. void f (int a[], int size)
B. int g (const int a[], int size)
C. void h (const int a[], int size)
D. int k (int a[], int size)

Which function prints a?

A. void f (int a[], int size)
B. int g (const int a[], int size)
C. void h (const int a[], int size)
D. int k (int a[], int size)
Which function returns the max value in a?

A. void f (int a[], int size)
B. int g (const int a[], int size)
C. void h (const int a[], int size)
D. int k (int a[], int size)

Which function squares all the values in a?

A. void f (int a[], int size)
B. int g (const int a[], int size)
C. void h (const int a[], int size)
D. int k (int a[], int size)
Clean Living

- Just as we avoid mixing value-returning functions with side-effects via call-by-reference
- Generally a function shouldn’t both update array elements and return a value.
- So `int k (int a[], int size)` is not recommended, as it allows side-effects on a and returns a value.

Exercises

- Write a function that reverses an array.
  - It takes two arrays the same size; the first contains integers, the second is empty. The function fills the second array with the elements of the first but in reverse order.
- Write a `search` function to find the position of a given value in an array.
  - This is a very useful function.
- Write a function that returns the index of the largest element of an array.
Count occurrences of integers

- Count how many of each integer 0 to 99 appears in a sequence of input.
  - Input ends with any number outside that range => while loop
  - How many counters do you need?
  - Data type of counters?
  - Initialise all counters
  - Which counter do you increment when you read in integer v?

Count votes

- Extend the idea of counting things.
- “Mary Jane Bob Bob Mary Bob Jane Bob Mary Mary Bob Dan ....... END”
- When counting non-integers you need to keep track of which counter corresponds to which thing being counted.
  - We use parallel arrays for this.
  - Look up the position of each name in the name array; increment that position in the counters.
Count votes in an input


<table>
<thead>
<tr>
<th>names</th>
<th>Mary</th>
<th>Jane</th>
<th>Bob</th>
<th>Dan</th>
<th>Joe</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>votes</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>0</td>
</tr>
</tbody>
</table>