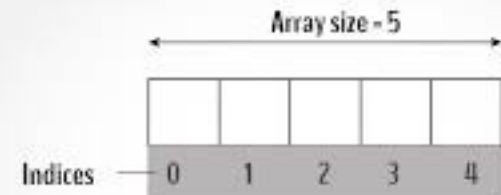


(8-1) Arrays I

H&K Chapter 7

Instructor – Beiyu Lin
CptS 121 (May 23rd, 2019)
Washington State University

What is an array?



C Arrays

Graphs are from:
<https://michaelscodingspot.com/array-iteration-vs-parallelism-in-c-net/>
<https://www.programiz.com/c-programming/c-arrays>



What is an array?

- A sequence of items that are contiguously allocated in memory
- All items in the array are of the same data type and of the same size
- All items are accessed by the same name, but a different index



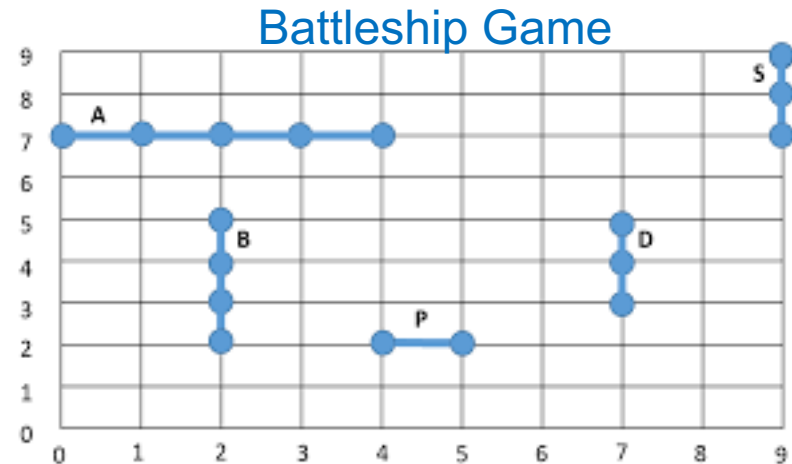
More About Arrays

- An array is a data structure
 - A data structure is a way of storing and organizing data in memory so that it may be accessed and manipulated efficiently



Uses for Arrays?

- Store related information
 - Student ID numbers
 - Names of players on the Seattle Mariners roster
 - Scores for each combination in Yahtzee



Graphs are from:

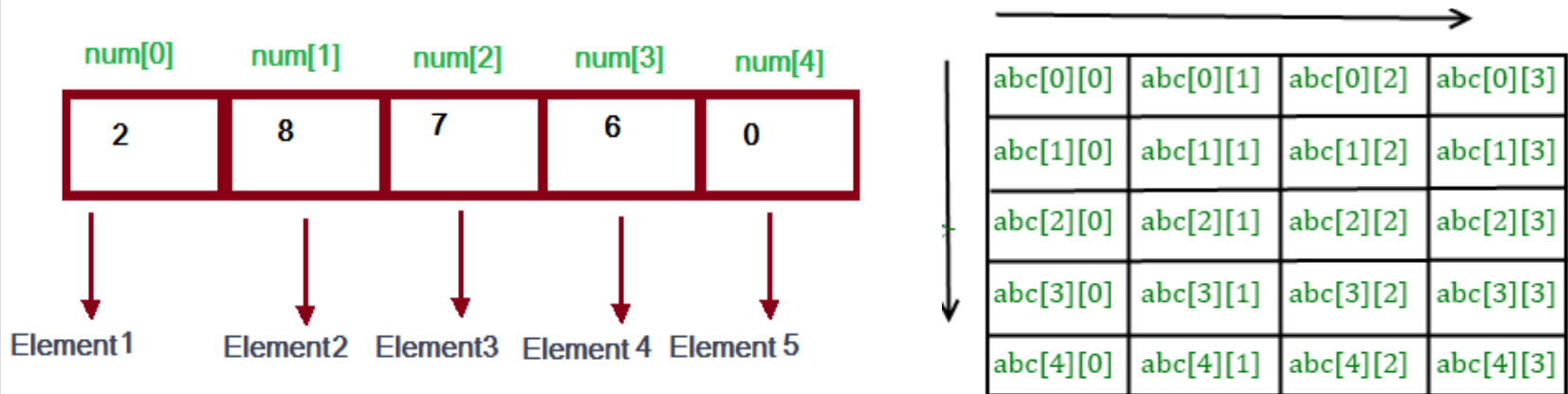
<https://www2.cs.arizona.edu/classes/cs120/fall17/ASSIGNMENTS/assg06/example-battleship.html>

<https://www.thoughtco.com/probability-of-rolling-a-yahtzee-3126593>



The Many Dimensions of an Array

- A single dimensional array is logically viewed as a linear structure
- A two dimensional array is logically viewed as a table consisting of rows and columns



Graphs are from: <https://study.com/academy/lesson/declaring-one-dimensional-arrays-definition-example.html>
<https://beginnersbook.com/2014/01/2d-arrays-in-c-example/>



Declaring a Single Dimensional Array (1)

- Arrays are declared in much the same way as variables:

```
int a[6];
```

declares an array `a` with 6 cells that hold integers:

<code>a[0]</code>	<code>a[1]</code>	<code>a[2]</code>	<code>a[3]</code>	<code>a[4]</code>	<code>a[5]</code>
10	12	0	89	1	91

Notice that array indexing begins at 0.



Declaring a Single Dimensional Array (2)

- We can declare arrays alongside simple variables:

```
int students[100], count, teachers[50];  
double gpa[100], average;  
char ch, name[100]; /* name is actually a string */
```



Manipulating Array Cells

- Assuming the previous array:

a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
10	12	0	89	1	91

all of the following statements are valid:

```
a[0] = 4; /* changes the value of a[0] from 10 to 4 */  
a[2] += 2; /* sets the value of a[2] to 2 */  
a[5] = a[3] - a[4]; /* sets the value of a[5] to 88 */
```



Initializing Arrays

- We can initialize arrays at the time we declare them
Just as

```
int count = 0;
```

is valid, so too is

```
int student_id[] = {3423, 8794, 4595, 1423,  
4311,  
5153, 9182, 1481, 1253,  
1222,  
2521, 2251, 2111};
```

Notice how you can omit the size of the array; the compiler deduces the size from the number of values listed.



Array Subscripts

- We can do arithmetic on array subscripts! Assume this array:

a[0] a[1] a[2] a[3] a[4] a[5]

10	12	0	89	1	91
----	----	---	----	---	----

Then all of the following are valid:

```
int x = 2;
printf("%d",a[x + 2]);      /* a[4] == 1 */
printf("%d",a[2 * x - 1]);  /* a[3] == 89 */
printf("%d",a[x] - a[x-1]); /* -12 */
printf("%d",a[++x]);        /* a[3] == 89; x == 3 */
a[x - 1] = a[x - 2];        /* assigns 12 to a[2] */
printf("%d",a[x + 4]);      /* Does a[7] exist? */
```



You Try It (1)

Write a segment of code that creates an array of 10 double values, populates the array with the values 1.0 through 10.0, and finally exchanges the 1st and 10th values.



You Try It (2)

Solution:

```
double array[] = {1.0, 2.0, 3.0, 4.0, 5.0, 6.0  
                  7.0, 8.0, 9.0, 10.0};  
  
double temp;  
temp = array[9];  
array[9] = array[0];  
array[0] = temp;
```



Using Loops to Access Array Elements (1)

- We often need to process each element of an array in turn
 - Example: Computing the average, minimum, and maximum of a group of values (sound familiar?)
- We can accomplish this with a `for` loop that goes from 0 to one less than the array size



Find Minimum Review

```
int num1= 2, num2=3, num3=10, num4=1;
int temp = num1;
if (num2 < temp)
{
    temp = num2;
}
If (num3< temp)
{
    temp = num3;
}
If (num4 < temp)
{
    temp = num4;
}
```



Using Array

```
int nums[] = {2,3,10,1};
min_num = nums[0];
int i = 0;
length_arr= sizeof(nums)/sizeof(int);

for (i = 0; i<length_arr; i ++)
{
    if (nums[i] < min_num)
    {
        min_num = nums[i];
    }
}
```



Using Loops to Access Array Elements (2)

```
int scores [] = {56,78,12,90,85,74,95,80,40,95};
int count = 10, i, sum = 0, max = 0, min = 100;
double average;
for (i = 0; i < count; ++i) /* we loop from 0 to 9 */
{
    sum += scores[i];
    if (scores[i] > max)
        max = scores[i];
    if (scores[i] < min)
        min = scores[i];
}
average = (double) sum / (double) count;
printf("average: %.2f\n",average);
printf("maximum: %d\n",max);
printf("minimum: %d\n",min);
/* Could also display a differences table here, just as
   the book does (see Fig. 8.3, p. 377 */
```



Passing Arrays as Parameters

- The previous example would exhibit better top-down design if it broke the problem down into functions:

- `get_scores` /* Let's assume that the scores should be read from an input file */
- `compute_stats` /* Given an array of values, computes the high, low, and average */
- `display_stats` /* Displays the high, low, and average */
- `display_differences_table` /* displays a table of the values read in and the difference between each value and the mean */



Next Lecture...

- We'll continue our exploration of arrays:
 - Searching and sorting algorithms
 - Multidimensional arrays



References

- J.R. Hanly & E.B. Koffman, *Problem Solving and Program Design in C (8th Ed.)*, Addison-Wesley, 2016
- P.J. Deitel & H.M. Deitel, *C How to Program (7th Ed.)*, Pearson Education , Inc., 2013.



Collaborators

- Chris Hundhausen
- Andrew O' Fallon

