CMPE 252
C PROGRAMMING

SPRING 2021

WEEK 5-6

ARRAY POINTERS
CHAPTER 7

Problem Solving & Program Design in C

Eighth Edition
Global Edition

Jeri R. Hanly & Elliot B. Koffman

Basic Terminology

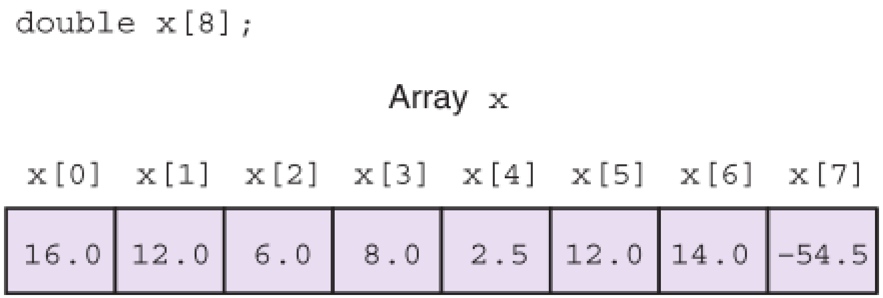
- data structure
  - a composite of related data items stored under the same name
- array
  - a collection of data items of the same type

Declaring and Referencing Arrays

- array element
  - a data item that is part of an array
- subscripted variable
  - a variable followed by a subscript in brackets, designating an array element
- array subscript
  - a value or expression enclosed in brackets after the array name, specifying which array element to access

FIGURE 7.1

The Eight Elements of Array x



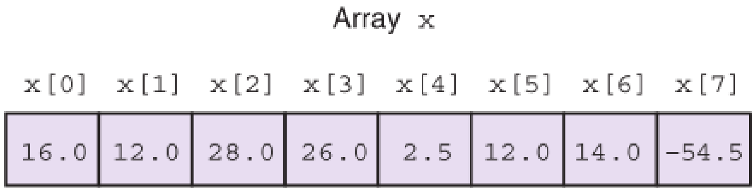
```
double x[8];
```

Array x

x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]
16.0	12.0	6.0	8.0	2.5	12.0	14.0	-54.5

TABLE 7.1 Statements That Manipulate Array x

Statement	Explanation
`printf("%.1f", x[0]);`	Displays the value of **x[0]**, which is **16.0**.
`x[3] = 25.0;`	Stores the value **25.0** in **x[3]**.
`sum = x[0] + x[1];`	Stores the sum of **x[0]** and **x[1]**, which is **28.0** in the variable **sum**.
`sum += x[2];`	Adds **x[2]** to **sum**. The new **sum** is **34.0**.
`x[3] += 1.0;`	Adds **1.0** to **x[3]**. The new **x[3]** is **26.0**.
`x[2] = x[0] + x[1];`	Stores the sum of **x[0]** and **x[1]** in **x[2]**. The new **x[2]** is **28.0**.



Array x

x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]
16.0	12.0	28.0	26.0	2.5	12.0	14.0	-54.5

Declaration

- You can declare more than one array in a single type declaration
- double cactus[5], needle, pins[6];
- int factor[12], n, index;

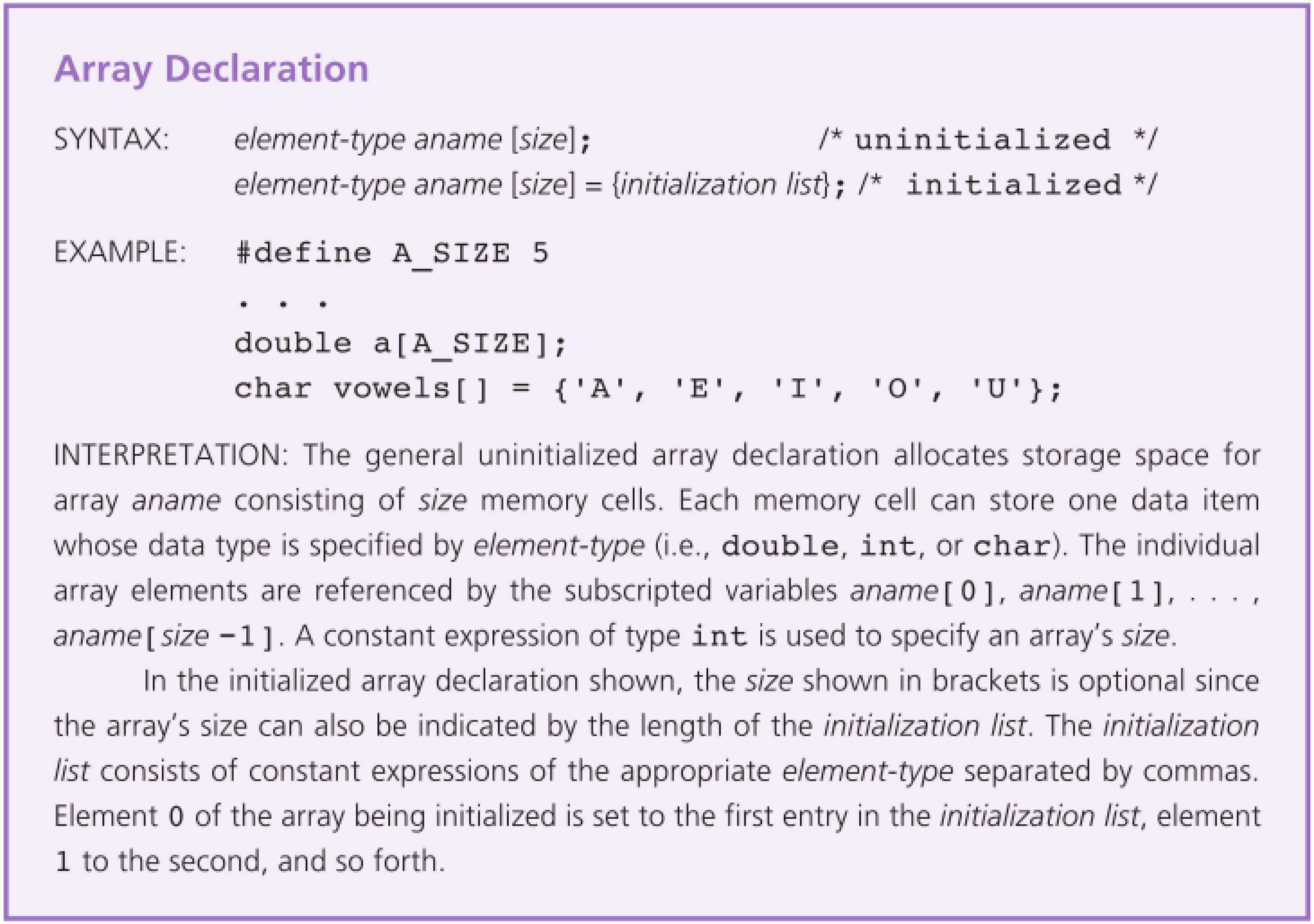
Array Initialization

- In declaration time, e.g.
- int prime_lt_100[] = {2, 3, 5, 7, 11, 13, 17, 19,
  23, 29, 31, 37, 41, 43, 47, 53, 59, 61,
  67, 71, 73, 79, 83, 89, 97}

- Size is deduced from the initialization list!!

- int x[5] = {10,20,30};

10	20	30	?	?



Array Declaration

SYNTAX: *element-type aname* [*size*]; /* **uninitialized** */
element-type aname [*size*] = {*initialization list*}; /* **initialized** */

EXAMPLE:

```
#define A_SIZE 5
. . .
double a[A_SIZE];
char vowels[] = {'A', 'E', 'I', 'O', 'U'};
```

INTERPRETATION: The general uninitialized array declaration allocates storage space for array *aname* consisting of *size* memory cells. Each memory cell can store one data item whose data type is specified by *element-type* (i.e., `double`, `int`, or `char`). The individual array elements are referenced by the subscripted variables *aname*`[0]`, *aname*`[1]`, . . . , *aname*`[`*size* `-1]`. A constant expression of type `int` is used to specify an array's *size*.

In the initialized array declaration shown, the *size* shown in brackets is optional since the array's size can also be indicated by the length of the *initialization list*. The *initialization list* consists of constant expressions of the appropriate *element-type* separated by commas. Element `0` of the array being initialized is set to the first entry in the *initialization list*, element `1` to the second, and so forth.

Storing a String in a Character Array

- char vowels[] = “hello”;

=

- char vowels[] = {‘h’,’e’,’l’,’l’,’o’,’\0’};

- Escape sequence is string termination null character \0
- Details will come in the following weeks about characters and strings!!

Array Subscripts

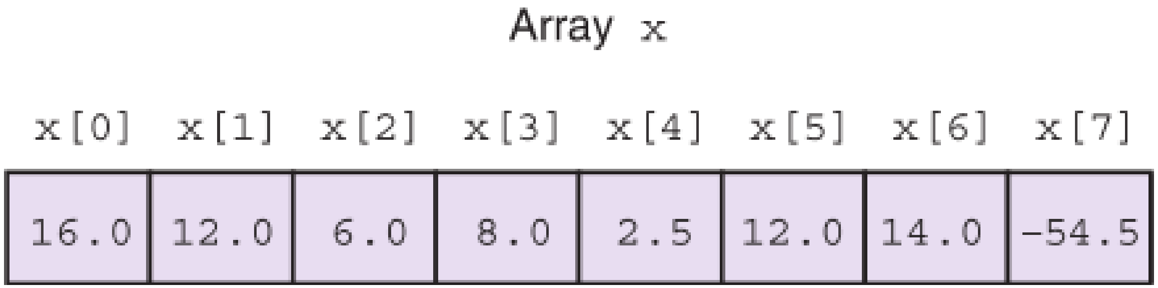
- Syntax:

  aname [subscript]

- Examples:

  x[3]

  x[i + 1]



Array x

x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]
16.0	12.0	6.0	8.0	2.5	12.0	14.0	-54.5

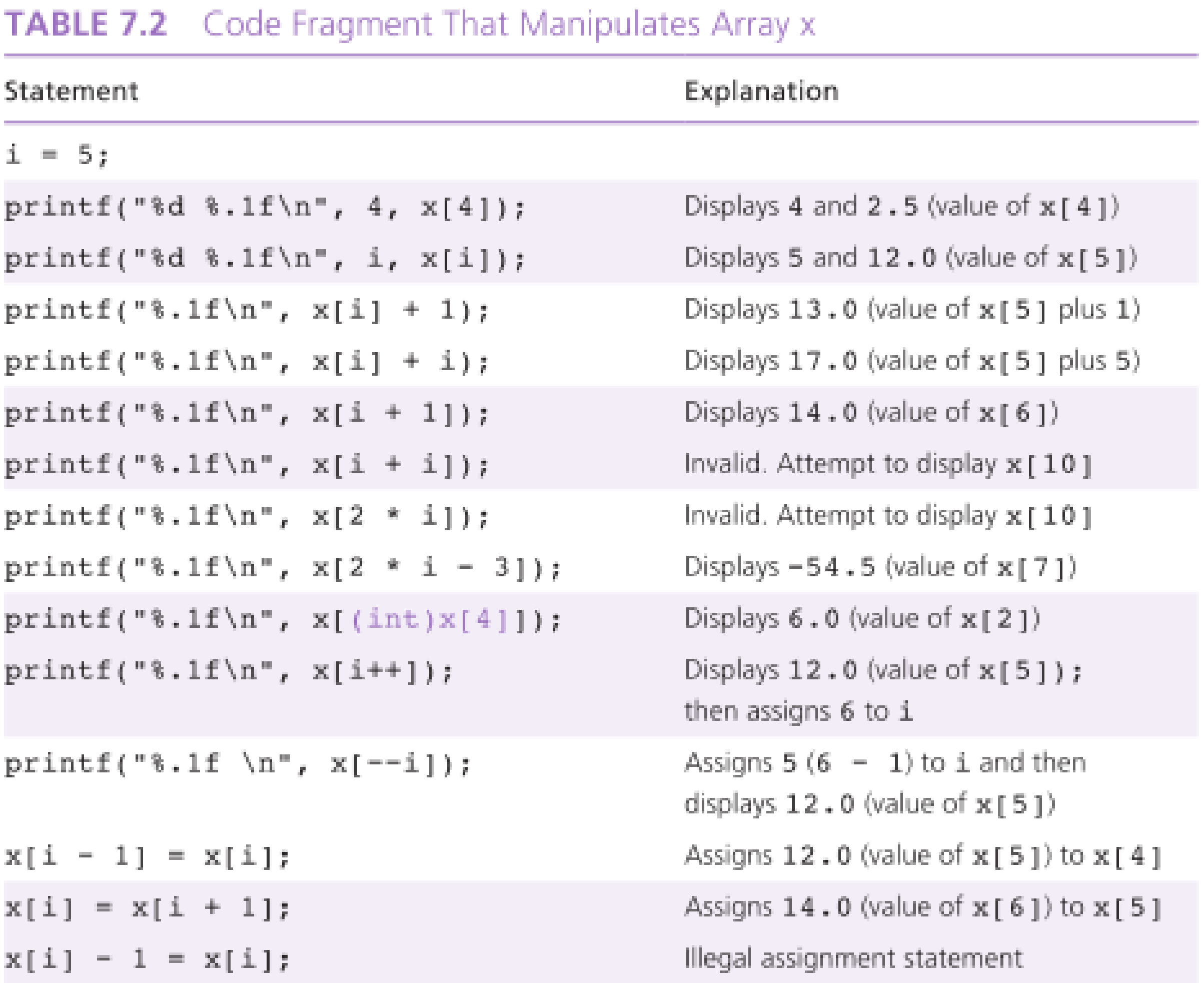
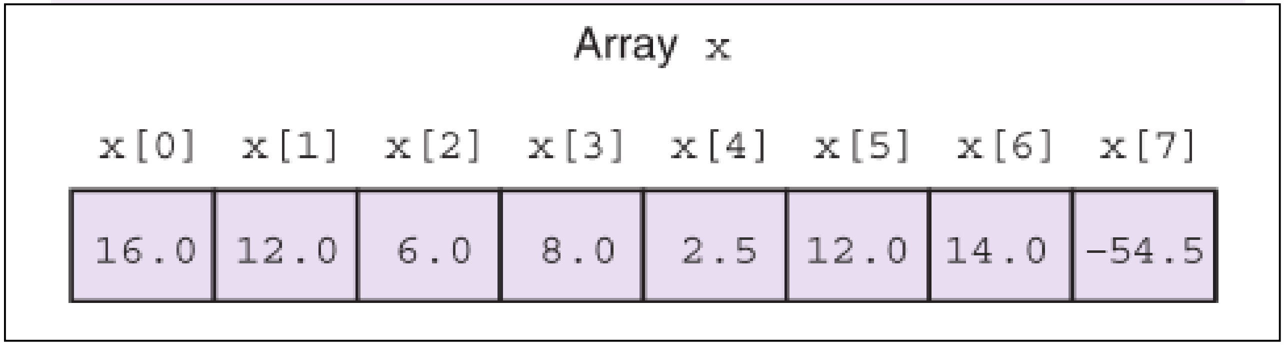


TABLE 7.2 Code Fragment That Manipulates Array x

Statement	Explanation
`i = 5;`	
`printf("%d %.1f\n", 4, x[4]);`	Displays 4 and 2.5 (value of x[4])
`printf("%d %.1f\n", i, x[i]);`	Displays 5 and 12.0 (value of x[5])
`printf("%.1f\n", x[i] + 1);`	Displays 13.0 (value of x[5] plus 1)
`printf("%.1f\n", x[i] + i);`	Displays 17.0 (value of x[5] plus 5)
`printf("%.1f\n", x[i + 1]);`	Displays 14.0 (value of x[6])
`printf("%.1f\n", x[i + i]);`	Invalid. Attempt to display x[10]
`printf("%.1f\n", x[2 * i]);`	Invalid. Attempt to display x[10]
`printf("%.1f\n", x[2 * i - 3]);`	Displays -54.5 (value of x[7])
`printf("%.1f\n", x[(int)x[4]]);`	Displays 6.0 (value of x[2])
`printf("%.1f\n", x[i++]);`	Displays 12.0 (value of x[5]); then assigns 6 to i
`printf("%.1f \n", x[--i]);`	Assigns 5 (6 - 1) to i and then displays 12.0 (value of x[5])
`x[i - 1] = x[i];`	Assigns 12.0 (value of x[5]) to x[4]
`x[i] = x[i + 1];`	Assigns 14.0 (value of x[6]) to x[5]
`x[i] - 1 = x[i];`	Illegal assignment statement



Array x

x[0]	x[1]	x[2]	x[3]	x[4]	x[5]	x[6]	x[7]
16.0	12.0	6.0	8.0	2.5	12.0	14.0	-54.5

References

1. Problem Solving & Program Design in C, Jeri R. Hanly & Elliot B. Koffman, Pearson 8. Edition, Global Edition
2. C How to Program, Paul Deitel, Harvey Deitel. Pearson 8th Edition, Global Edition.
3. https://www.geeksforgeeks.org/bubble-sort/