

Data Structures Arrays and Structs

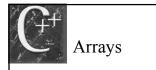
Chapter 9



9.1 The Array Data Type

- Array elements have a common name
 The array as a whole is referenced through the common name
- Array elements are of the same type the base type
- Individual elements of the array are referenced by sub_scripting the group name

2

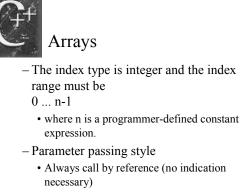


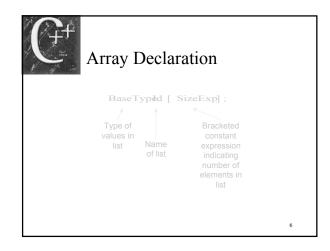
- Analogies
 - Egg carton
 - Apartments
 - Cassette carrier
- More terminology
 - Ability to refer to a particular element
 - Indexing or sub_scripting
 - Ability to look inside an element
 - Accessing value



Arrays

- Language restrictions
 - Subscripts are denoted as expressions within brackets: []
 - Base type can be any fundamental, library-defined, or programmer -defined type





Ċ

Sample Declarations

Suppose
 const int N = 20;
 const int M = 40;
 const int MaxStringSize = 80;
 const int MaxListSize = 1000;



Sample Declarations

Then the following are all correct array declarations.

 int A[10];
 char B[MaxStringSize];
 float C[M*N];
 int Values[MaxListSize];
 Rational D[N-15];

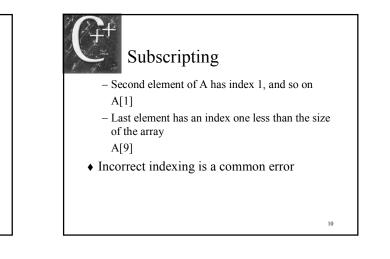


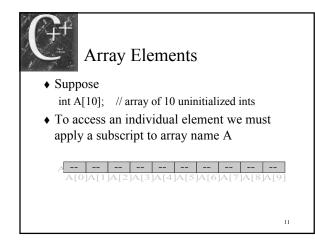
Subscripting

♦ Suppose

- int A[10]; // array of 10 ints
- To access an individual element we must apply a subscript to array name A
 - A subscript is a bracketed expression
 - The expression in the brackets is known as the index
 - First element of A has index 0

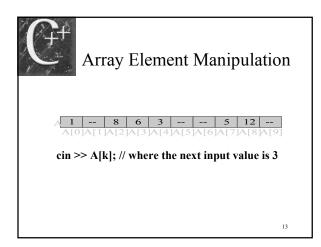
A[0]

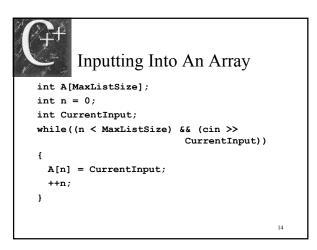


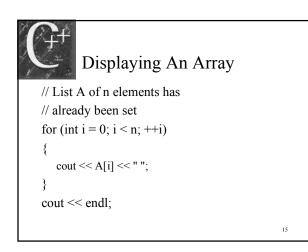


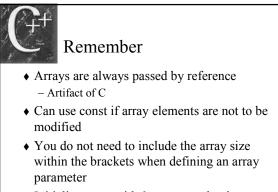
Array Element Manipulation

 ♦ Given the following: int i = 7, j = 2, k = 4; A[0] = 1; A[i] = 5; A[j] = A[i] + 3; A[j+1] = A[i] + A[0]; A[A[j]] = 12;









```
    Initialize array with 0 or some other known value
```

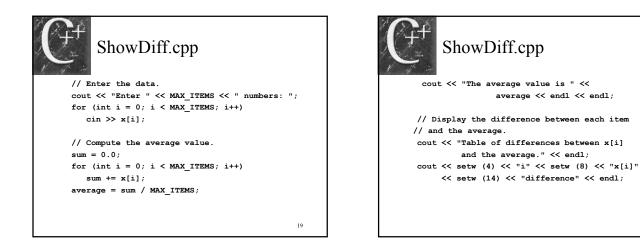


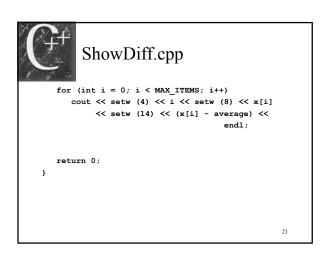
9.2 Sequential Access to Array Elements

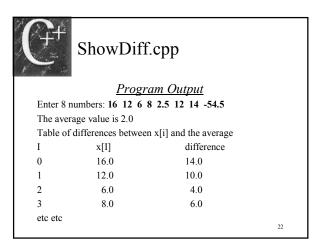
- ♦ Random Access
 - Access elements is random order
- Sequential Access
 - Process elements in sequential order starting with the first
 - ShowDiff.cpp a program that looks at values and calculates a difference between the element and the average

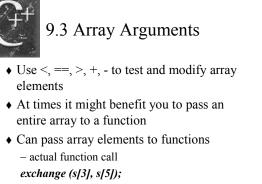
17

subsect by the set of the se

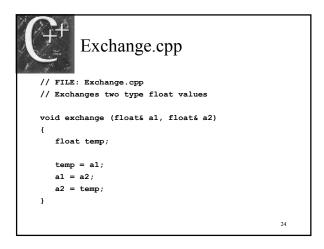








♦ Examples follow





Arrays as Function Arguments

- Remember arrays are pass by reference - Passing the array address
- Remember these points when passing arrays to functions
 - The formal array argument in a function is not itself an array but rather is a name that represents an actual array argument. Therefore in the function definition, you need only inform the compiler with [] that the actual argument will be an array 25



Arrays as Function Arguments

26

- Remember these points when passing arrays to functions
 - Formal array arguments that are not to be altered by a function should be specified using the reserved word *const*. When this specification is used, any attempt to alter the contents will cause the compiler generate an error message
- SameArray.cpp example

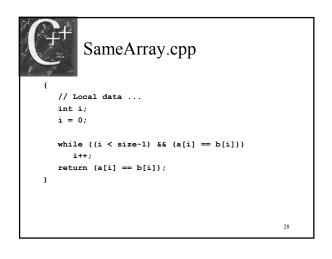


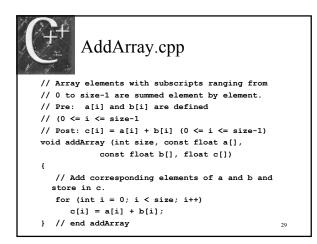
SameArray.cpp

- // FILE: SameArray.cpp
- // COMPARES TWO FLOAT ARRAYS FOR EQUALITY BY
- // COMPARING CORRESPONDING ELEMENTS
- // Pre: a[i] and b[i] (0 <= i <= size-1) are</pre>
- // assigned values.
- // Post: Returns true if a[i] == b[i] for all I
- // in range 0 through size 1; otherwise,
- // returns false.

bool sameArray (float a[], float b[], const int size)

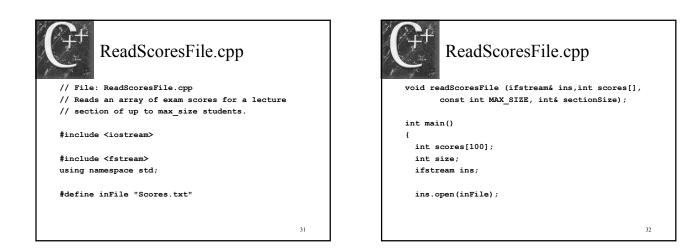


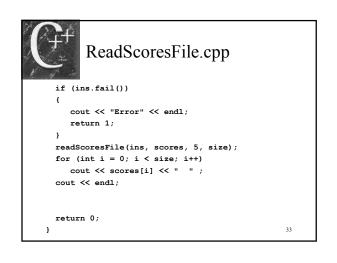


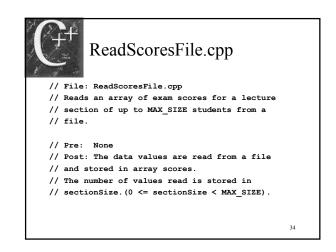


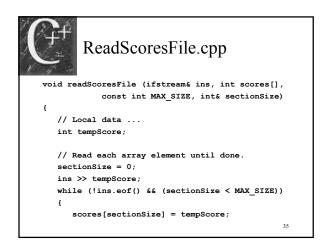
9.4 Reading Part of an Array • Sometimes it is difficult to know how many elements will be in an array ♦ Scores example - 150 students - 200 students

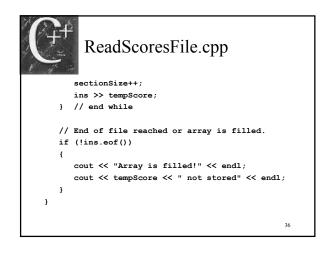
- Always allocate enough space at compile time
- Remember to start with index [0]













9.5 Searching and Sorting Arrays

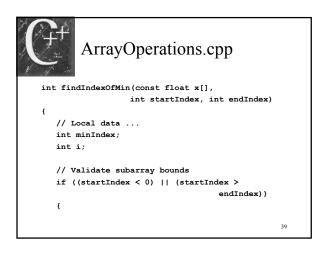
- Look at 2 common array problems
 Searching
 - Sorting
 - Sorting
- How do we go about finding the smallest number in an array?
 - Assume 1st is smallest and save its position
 - Look for one smaller
 - If you locate one smaller save its position

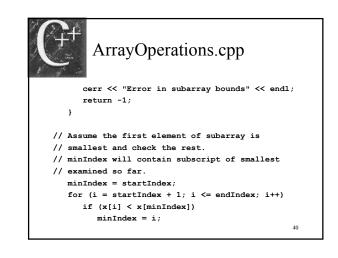


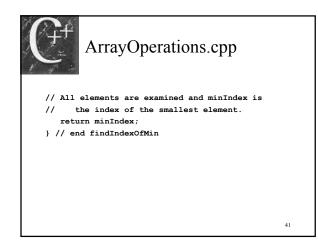
ArrayOperations.cpp

- // File: arrayOperations.cpp
- // Finds the subscript of the smallest value in a
 // subarray.
- // Returns the subscript of the smallest value
- // in the subarray consisting of elements
- // x[startindex] through x[endindex]
- // Returns -1 if the subarray bounds are invalid.
- // Pre: The subarray is defined and 0 <=</pre>
- // startIndex <= endIndex.</pre>
- // Post: x[minIndex] is the smallest value in
- // the array.

37







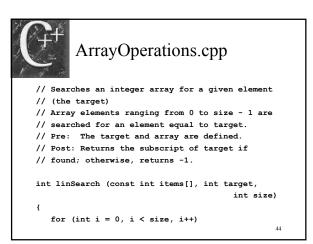


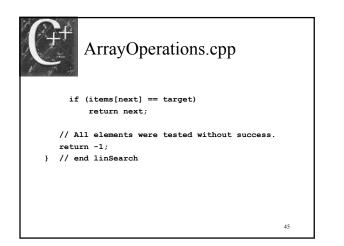
Strings and Arrays of Characters

- String object uses an array whose elements are type *char*
- First position of a string object is 0
 example string find function ret of position 0
- Can use the find function to locate or search an array
- We will study some various search functions

Linear Search

- The idea of a linear search is to walk through the entire until a target value is located
- If the target is not located some type of indicator needs to be returned







43

47

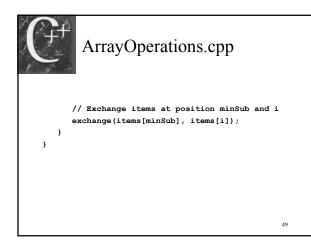
Sorting in Ascending Order Selection Sort

- Idea of the selection sort is to locate the smallest value in the array
- Then switch positions of this value and that in position [0]
- We then increment the index and look again for the next smallest value and swap

46

♦ Continue until sorted

// Sorts an array (ascending order) using
// selection sort algorithm
// Uses exchange and findIndexOfMin
// Sorts the data in array items (items[0]
// through items[n-1]).
// Pre: items is defined and n <= declared size
// of actual argument array.
// Post: The values in items[0] through items
// [n-1] are in increasing order.</pre>





9.7 Analyzing Algorithms Big O Notation

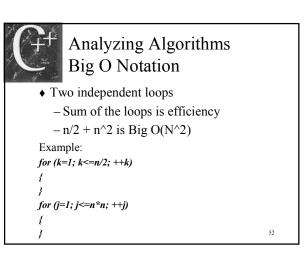
- How to compare efficiency of various algorithms
- A mathematical measuring stick to do quantitative analysis on algorithms
- Typically sorting and searching
- Based on looping constructs and placed into categories based on their efficiency
- Most algorithms have BigO published

50



Analyzing Algorithms Big O Notation

- Run time efficiency is in direct proportion to the number of elementary machine operations
 - Compares
 - Exchanges



Analyzing Algorithms Big O Notation



51

53

9.7 The Struct Data Type

- struct used to store related data items
- Individual components of the struct are called its members
- Each member can contain different types of data
- ♦ Employee example

Cf ⁺ Struct Employee	
<pre>// Definition of struct employee</pre>	
<pre>struct employee { string id;</pre>	
string name;	
-	
char gender;	
int numDepend;	
money rate;	
<pre>money totWages;</pre>	
};	
	55



Accessing Members of a struct

- Members are accessed using the *member* access operator, a period
- For struct variable s and member variable m to access m you would use the following: - cout << s.m << endl;
- Can use all C++ operators and operations on structs



Accessing Members of a struct

organist.id = 1234; organist.name = "Noel Goddard"; organist.gender = 'F'; organist.numDepend = 0; organist.rate = 6.00; organist.totWages += organist.rate * 40.0;



9.8 Structs as Operands and Arguments

56

58

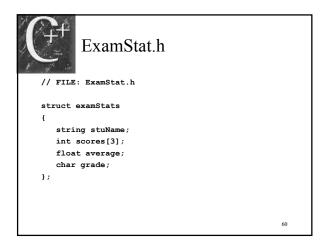
- How to do arithmetic and other operations using structs
- Process entire struct using programmer defined functions
- Often better to pass an entire structure rather than individual elements
- struct copies
 organist = janitor;

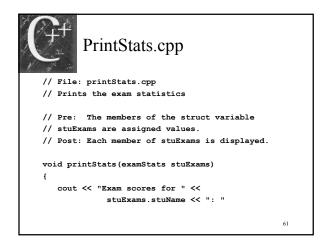


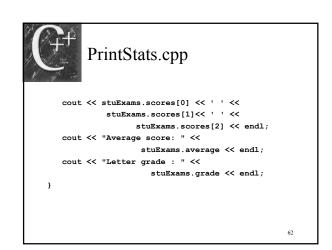
Passing struct as an Argument

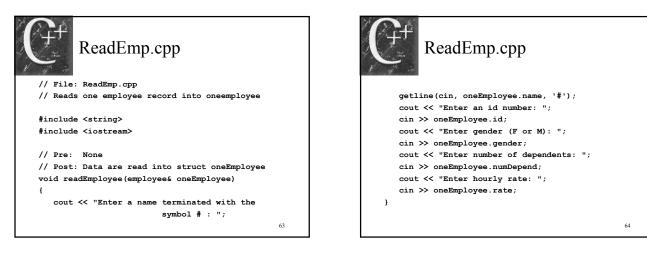
- Grading program example
- Keep track of students grades
- Prior to our learning structs we needed to store each item into a single variable
- Group all related student items together
- Pass struct by *const* reference if you do not want changes made

59











9.9 Common Programming Errors

- Watch non int subscripts (ASCII value)
- Enumerated types can be used
- ♦ Out of range errors
- C++ no range error checking
- Lack of subscript to gain access
- Subscript reference to non-array variable
- Type mixing when using with functions
- ♦ Initialization of arrays

65

C^+

Common Programming Errors

- No prefix to reference a struct member
- Incorrect prefix reference to a struct member
- Missing ; following definition of struct
- Initialization of struct members