Data Structure and Algorithms

Affefah Qureshi

Department of Computer Science

Iqra University, Islamabad Campus.

Arrays

- An array is defined as
  - Ordered collection of a fixed number of elements
  - All elements are of the same data type

- Basic operations
  - Direct access to each element in the array
  - Values can be retrieved or stored in each element

Properties of an Array

- Ordered
  - Every element has a well defined position
  - First element, second element, etc.
- Fixed size or capacity
  - Total number of elements are fixed
- Homogeneous
  - Elements must be of the same data type (and size)
  - Use arrays only for homogeneous data sets
- Direct access
  - Elements are accessed directly by their position
  - Time to access each element is same
  - Different to sequential access where an element is only accessed after the preceding elements

Declaring Arrays in C/C++

dataType arrayName[size];

- datatype – Any data type, e.g., integer, character, etc.
- arrayName – Name of array using any valid identifier
- intExp – Constant expression that evaluates to a positive integer

Why constant?

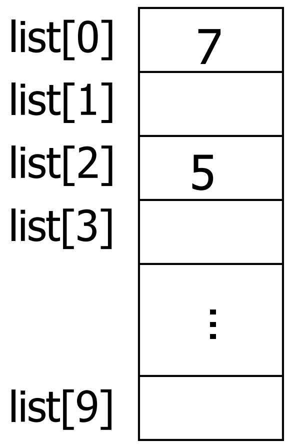
- Example:
  - const int SIZE = 10;
  - Int list[SIZE];
- Compiler reserves a block of consecutive memory locations enough to hold size of values int.

IU

Accessing Arrays in C/C++

arrayName[index];

- indexExp – called index, is any expression that evaluates to a positive integer
- In C/C++
  - Array index starts at 0
  - Elements of array are indexed 0, 1, 2, ..., SIZE-1
  - [ ] is called array subscripting operator
- Example
  - Int value = list[2];
  - list[0] = value + 2;



C/C++ Implementation of an Array ADT

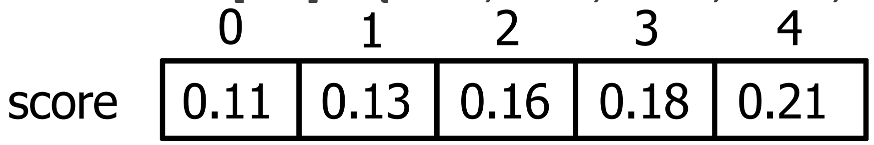
As an ADT	**In C/C++**
Ordered	Index: `0,1,2, … SIZE-1`
Fixed Size	`intExp` is constant
Homogeneous	dataType is the type of all elements
Direct Access	Array subscripting operator [ ]

Array Initialization in C/C++

dataType arrayName[index] = {list of values}

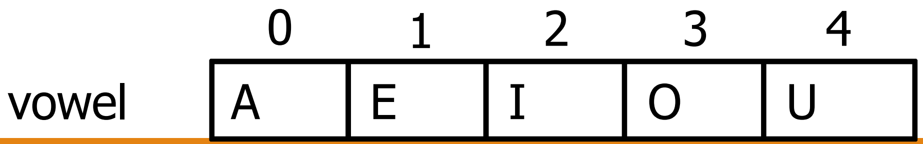
- In C/C++, arrays can be initialized at declaration
- index is optional: Not necessary to specify the size
- Example: Numeric arrays

  double score[ ] = {0.11,0.13, 0.16, 0.18, 0.21}



- Example: Character arrays

  char vowel[5] = { ‘A’, ‘E’, ‘I’, ‘O’, ‘U’ }



Array Initialization in C/C++

- Fewer values are specified than the declared size of an array
  - Numeric arrays: Remaining elements are assigned zero
  - Character arrays: Remaining elements contains null character ‘\0’
    - ASCII code of ‘\0’ is zero
- Example:
- double score[5] = {0.11, 0.13, 0.16}

	0	1	2	3	4
score	0.11	0.13	0.16	0	0

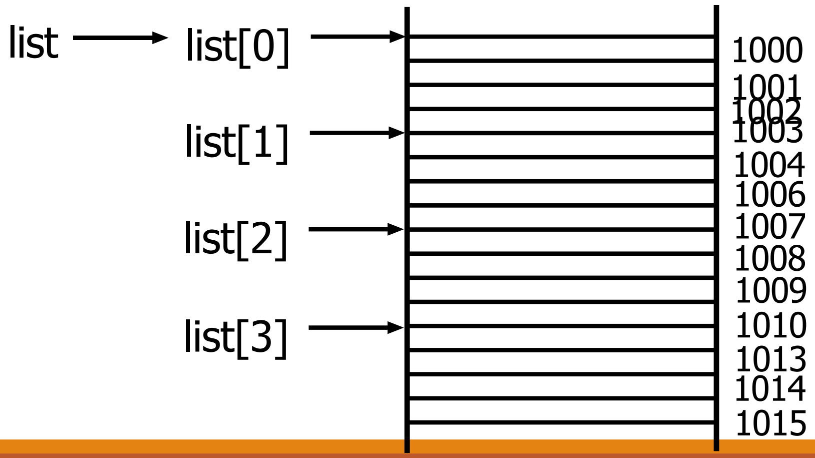
- char name[6] = {‘J’, ‘O’, ‘H’, ‘N’}

	0	1	2	3	4	5
name	J	O	H	N	\0	\0

- If more values are specified than declared size of an array
  - Error is occurred: Handling depends on compiler

Array Addressing

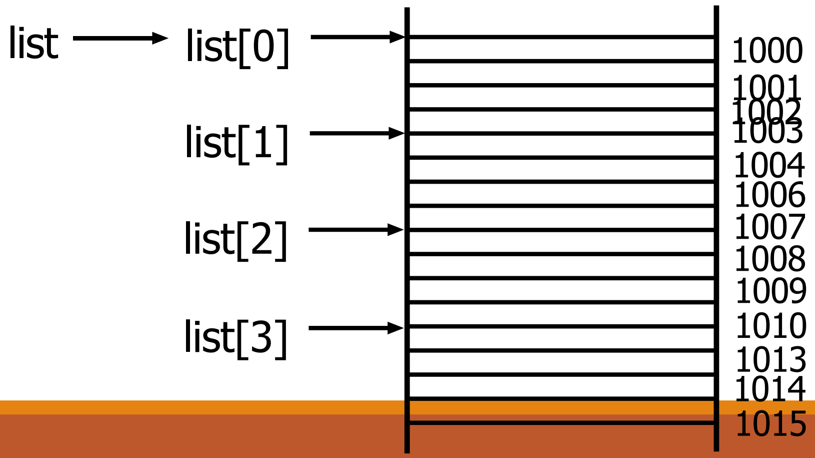
- Consider an array declaration: int list [4] = { 1, 2, 4, 5}
  - Compiler allocates a block of four memory spaces
  - Each memory space is large enough to store an int value
  - Four memory spaces are contiguous
- Base address
  - Address of the first byte (or word) in the contiguous block of memory
  - Address of the memory location of the first array element
    - Address of element list[0]
- Memory address associated with arrayName stores the base address
- Example:
  - cout << list << endl; (Print 1000)
  - cout << *list << endl; (Print 1)
- * is dereferencing operator
  - Returns content of a memory location



Array Addressing

- Consider a statement: cout << list[3];
  - Requires array reference list[3] be translated into memory address
  - Offset: Determines the address of a particular element w.r.t. base address
- Translation
  - Base address + offset = 1000 + 3 x sizeof(int) = 1012
  - Content of address 1012 are retrieved & displayed
- An address translation is carried out each time an array element is accessed
- What will be printed and why?

  cout << *(list+3) << endl;



Questions

Why does an array index start at zero?

Why arrays are not passed by value?

Two Dimensional Arrays – Declaration

dataType arrayName[intExp1][intExp2];

- intExp1 – constant expression specifying number of rows
- intExp2 – constant expression specifying number of columns
- Example:

  const int NUM_ROW = 2, NUM_COLUMN = 4;

  double scoreTable [NUM_ROW][NUM_COLUMN];
- Initialization:

  Double scoreTable [ ][4] = {{0.5, 0.6, 0.3}, {0.6, 0.3, 0.8}};
  - List the initial values in braces, row by row
  - May use internal braces for each row to improve readability

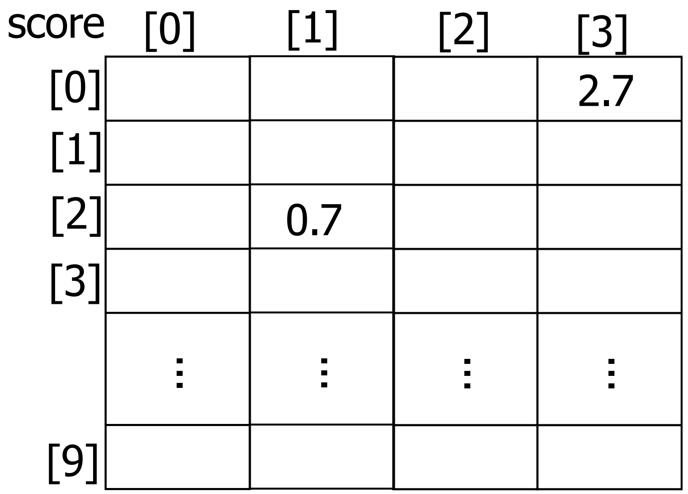
Two Dimensional Arrays – Processing

arrayName[row index][col index];

- indexExp1 – row index
- indexExp2 – column index

- Rows and columns are numbered from 0
- Use nested loops to vary two indices
  - Row-wise or column-wise manner
- Example:

```
double   value = score[2][1];
score[0][3] = value + 2.0;
```



Two-dimensional Arrays in Memory

- Two ways to be represented in memory
  - Column majored
    - Column by column
  - Row majored
    - Row by row
  - Representation depends upon the programming language

	(1,1)	
	(2,1)	Column 1
	(3,1)	
	(1,2)	
	(2,2)	Column 2
	(3,2)	
	(1,3)	
	(2,3)	Column 3
	(3,3)	
	(1,4)	
	(2,4)	Column 4
	(3,4)	

	(1,1)	
	(1,2)	Row 1
	(1,3)	
	(1,4)	
	(2,1)	
	(2,2)	Row 2
	(2,3)	
	(2,4)	
	(3,1)	
	(3,2)	Row 3
	(3,3)	
	(3,4)	

Any Question So Far?