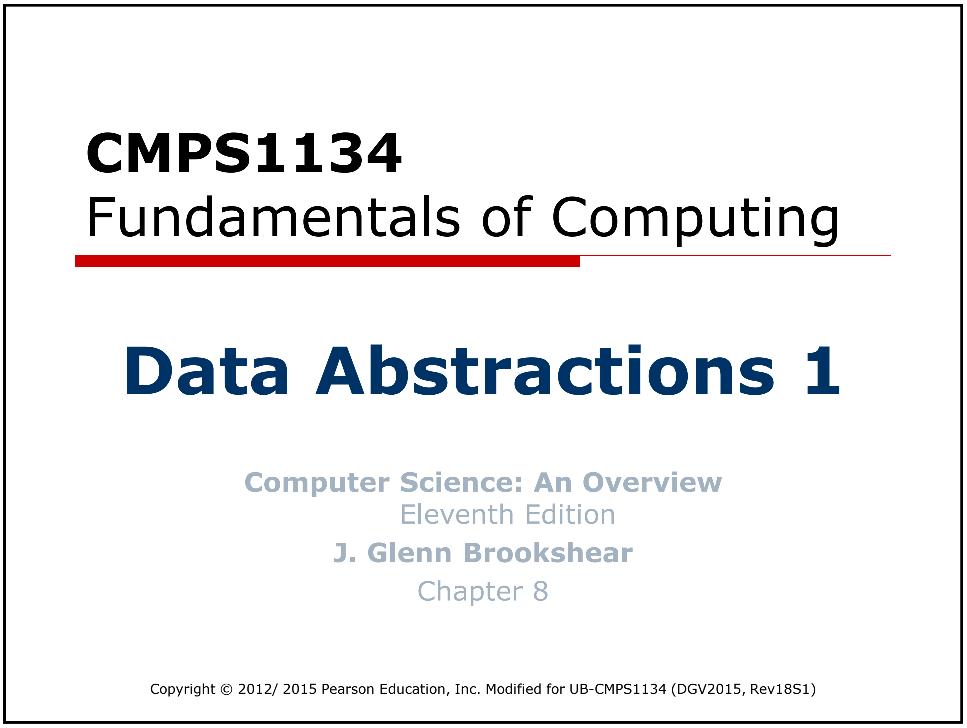
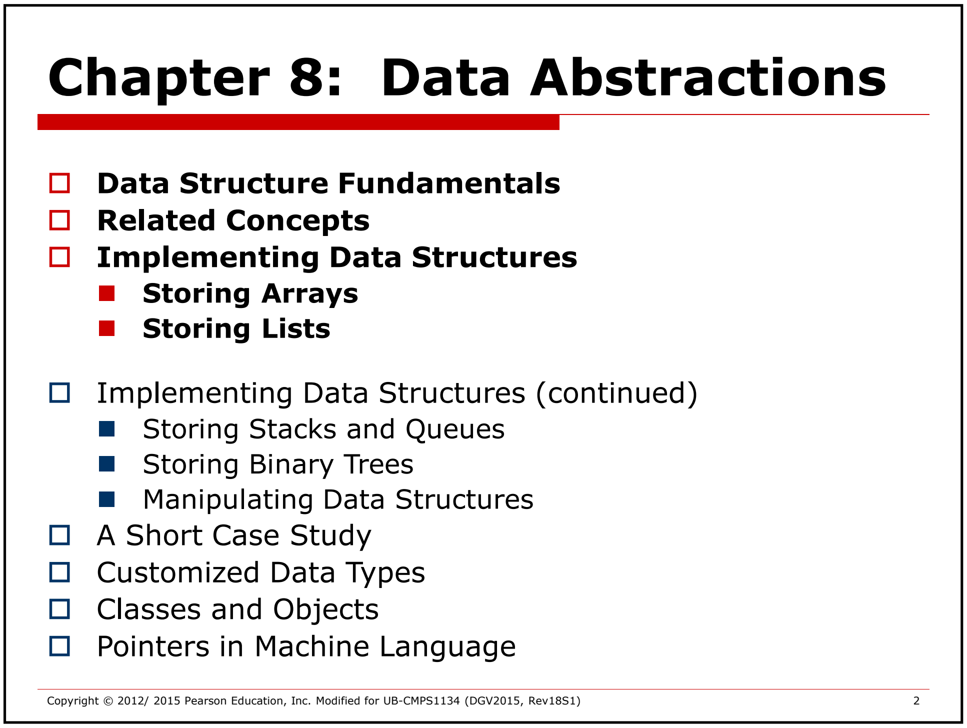


CMPS1134
Fundamentals of Computing

Data Abstractions 1

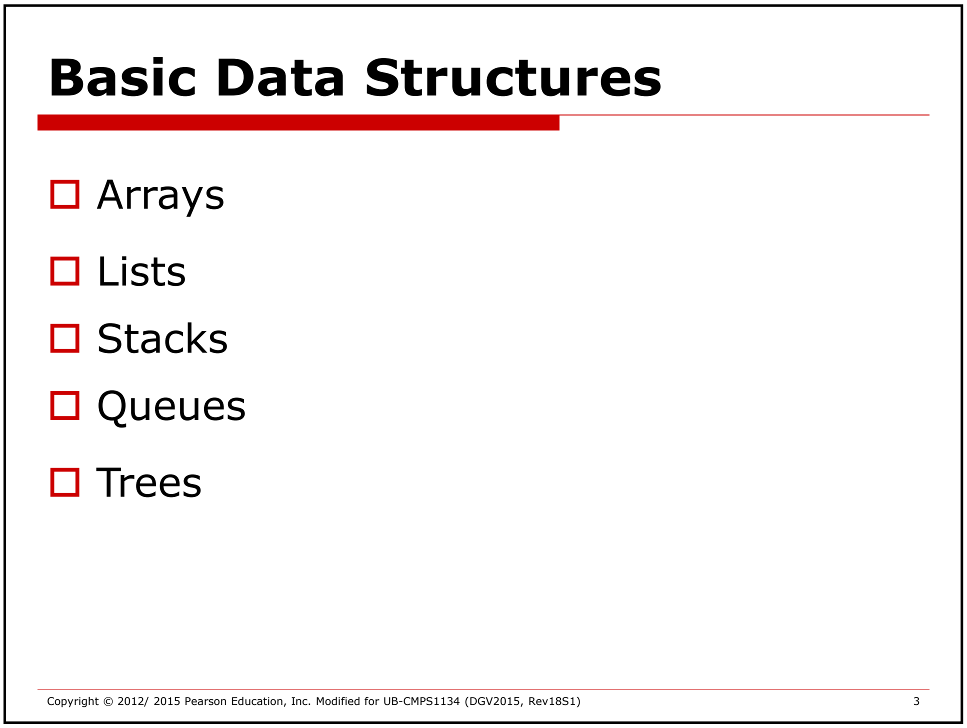
Computer Science: An Overview
Eleventh Edition
J. Glenn Brookshear
Chapter 8



Chapter 8: Data Abstractions

- **Data Structure Fundamentals**
- **Related Concepts**
- **Implementing Data Structures**
  - **Storing Arrays**
  - **Storing Lists**

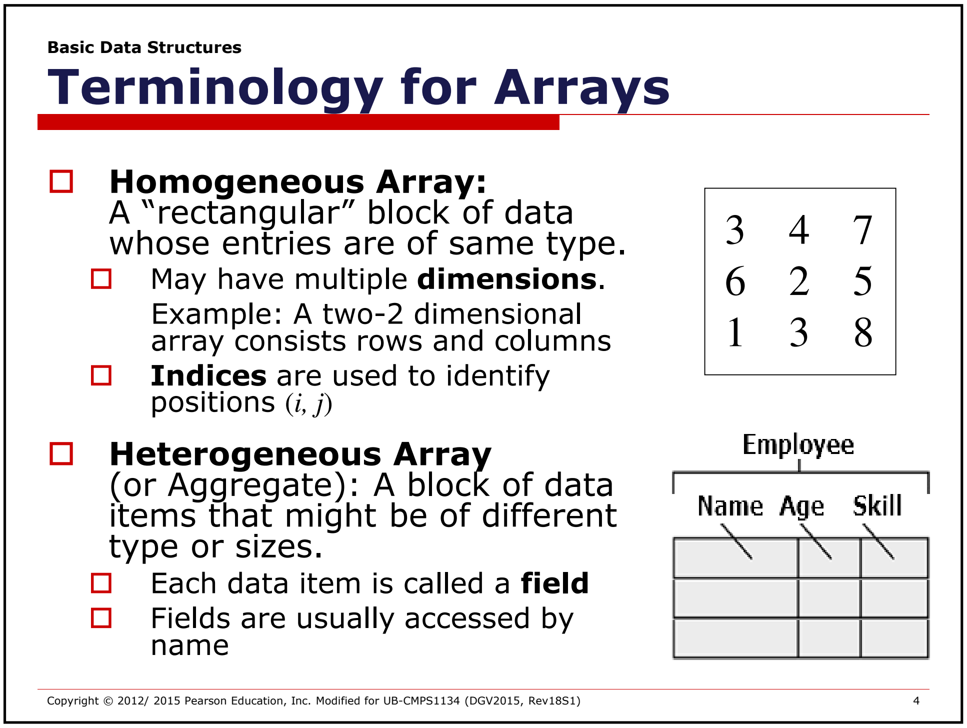
- Implementing Data Structures (continued)
  - Storing Stacks and Queues
  - Storing Binary Trees
  - Manipulating Data Structures
- A Short Case Study
- Customized Data Types
- Classes and Objects
- Pointers in Machine Language



Basic Data Structures

- □ Arrays
- □ Lists
- □ Stacks
- □ Queues
- □ Trees

3



Basic Data Structures

Terminology for Arrays

- □ **Homogeneous Array:** A "rectangular" block of data whose entries are of same type.
  - □ May have multiple **dimensions**. Example: A two-2 dimensional array consists rows and columns
  - □ **Indices** are used to identify positions (i, j)
- □ **Heterogeneous Array** (or Aggregate): A block of data items that might be of different type or sizes.
  - □ Each data item is called a **field**
  - □ Fields are usually accessed by name

3	4	7
6	2	5
1	3	8

Employee

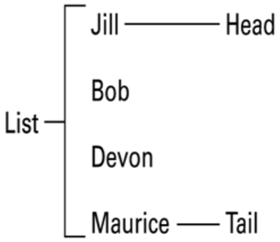
Name	Age	Skill

4

Basic Data Structures

Terminology for Lists

List:
A collection of data whose entries are arranged sequentially

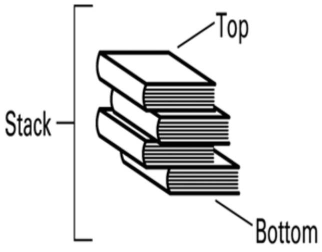


- ☐ **Head:** The beginning of the list
- ☐ **Tail:** The end of the list

Basic Data Structures

Terminology for Stacks

Stack:
A list in which entries are removed and inserted only at the head



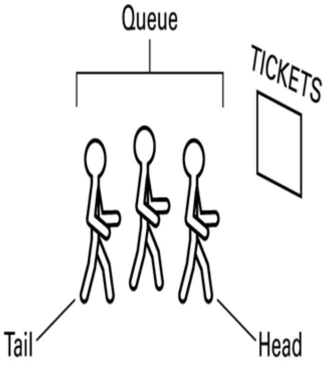
- ☐ **LIFO:** Last-in-first-out
- ☐ **Top:** The head of list (stack)
- ☐ **Bottom** or **base:** The tail of list (stack)
- ☐ **Pop:** To remove the entry at the top
- ☐ **Push:** To insert an entry at the top

Basic Data Structures

Terminology for Queues

Queue:
A list in which entries are removed at the head and are inserted at the tail

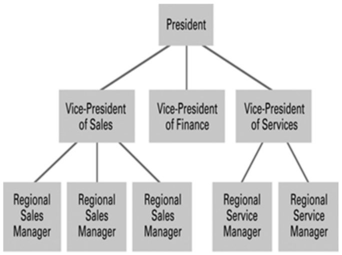
- □ **FIFO:** First-in-first-out



Basic Data Structures

Terminology for a Tree

Tree:
A collection of data whose entries have a hierarchical organization



President

Vice-President of Sales

Vice-President of Finance

Vice-President of Services

Regional Sales Manager

Regional Sales Manager

Regional Sales Manager

Regional Service Manager

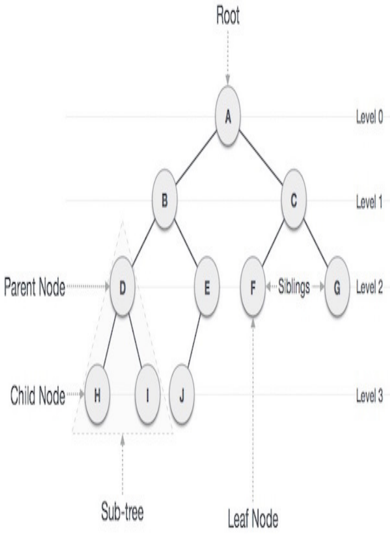
Regional Service Manager

- □ **Node:** An entry in a tree
- □ **Root node:** The node at the top
- □ **Terminal** or **leaf node:** A node at the bottom

Basic Data Structures

Terminology for a Tree (cont)

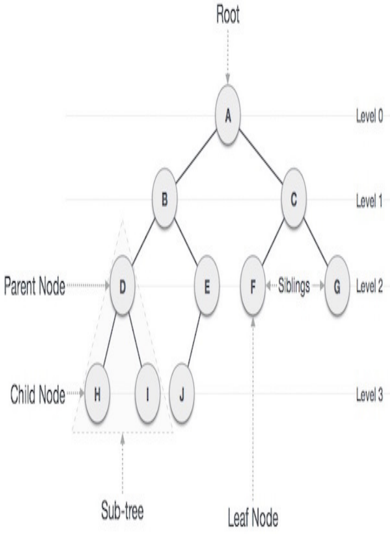
- □ **Parent:** The node immediately above a specified node
- □ **Child:** A node immediately below a specified node
- □ **Ancestor:** Parent, parent of parent, etc.
- □ **Descendent:** Child, child of child, etc.
- □ **Siblings:** Nodes sharing a common parent

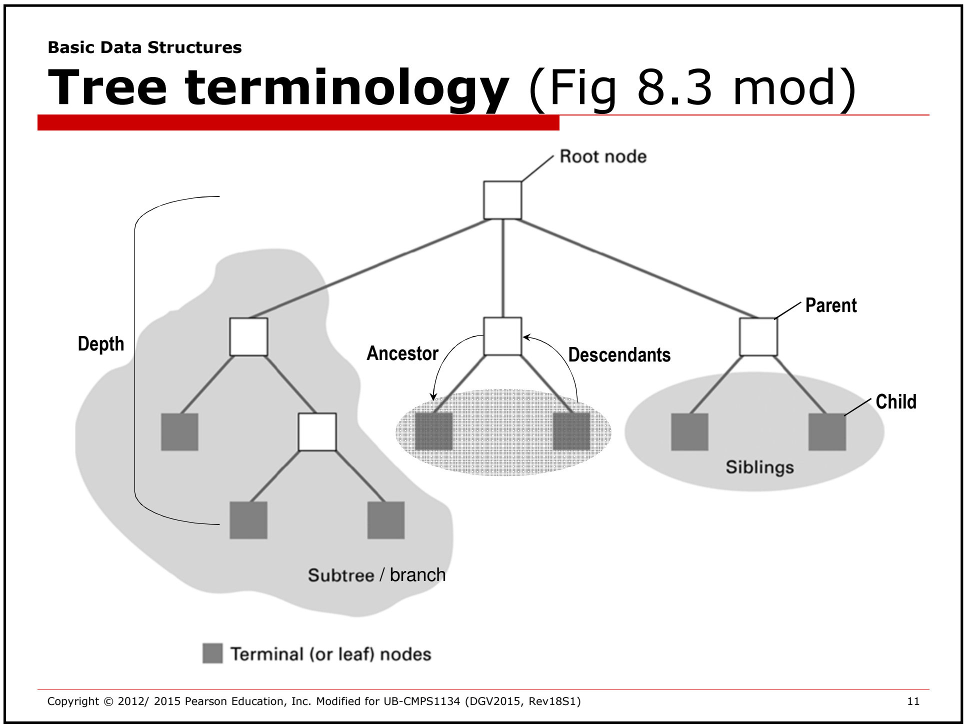


Basic Data Structures

Terminology for a Tree (cont)

- □ **Binary tree:** A tree in which every node has at most two children
- □ **Depth:** The number of nodes in longest path from root to leaf
- □ **Subtree:** A node together with all the nodes below it. Each subtree is a **branch** from its parent.

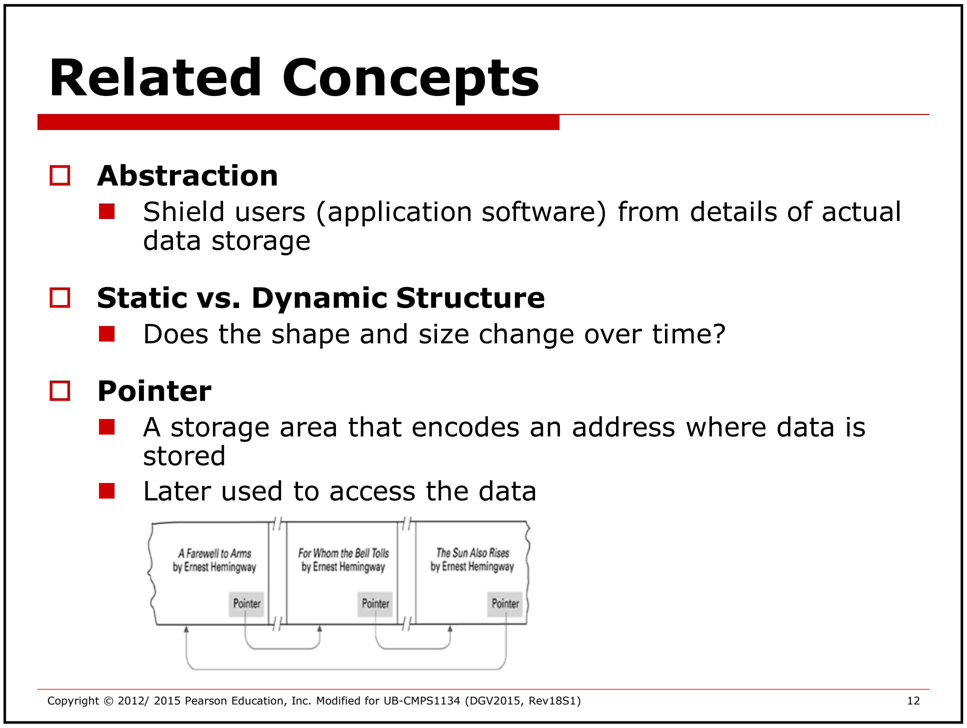




Basic Data Structures

Tree terminology (Fig 8.3 mod)

■ Terminal (or leaf) nodes



Related Concepts

- **Abstraction**
  - Shield users (application software) from details of actual data storage
- **Static vs. Dynamic Structure**
  - Does the shape and size change over time?
- **Pointer**
  - A storage area that encodes an address where data is stored
  - Later used to access the data

A Farewell to Arms by Ernest Hemingway
Pointer
For Whom the Bell Tolls by Ernest Hemingway
Pointer
The Sun Also Rises by Ernest Hemingway
Pointer

Implementing Data Structures

Different techniques are used for storing data structures in a computer's main memory.

We look at:

- □ **Storing Arrays** (homo/heterogeneous)
- □ **Storing Lists**
- □ Storing Stacks and Queues
- □ Storing Binary Trees
- □ Manipulating Data Structures

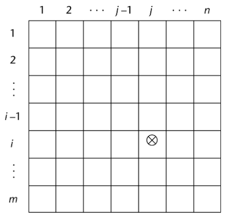
Implementing Data Structures

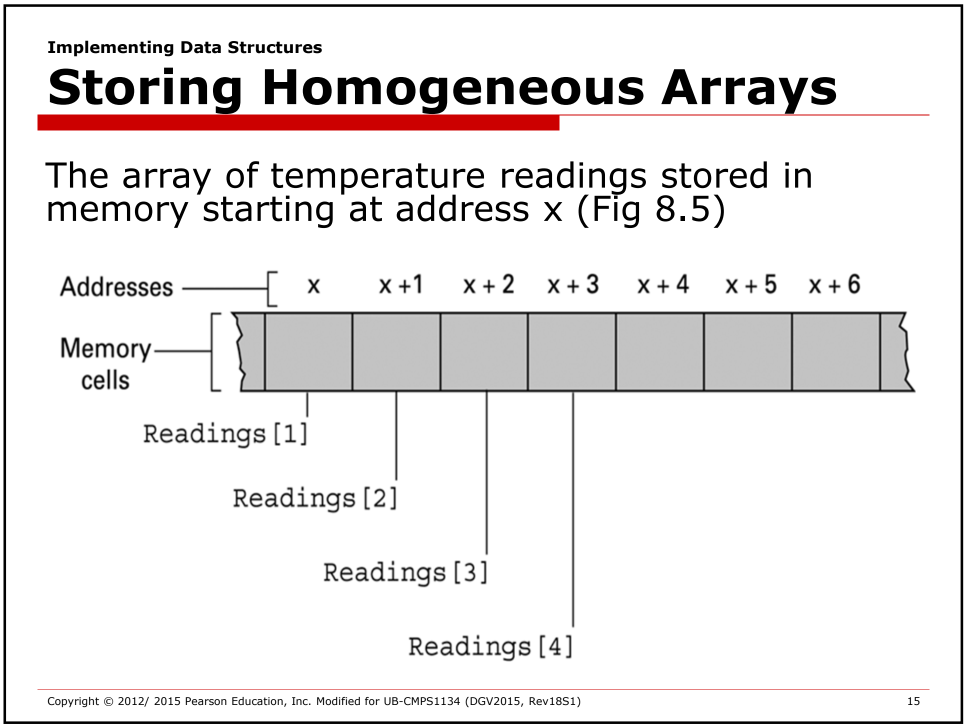
Storing Homogeneous Arrays

Memory address of a particular cell can be computed

- □ **Row-major order** versus **column major order**
  - Row major order (by rows) used in most languages: 3,4,7,6,2,5,1,3,8
  - Column major order (by columns) used in Fortran: 3,6,1,4,2,3,7,5,8
- □ Address polynomial
  - $(c \times (i\text{-}1)) + (j\text{-}1)$
  - Cell size (c) multiplied by the number of rows (i-1) above an element, plus the number of elements to the left of the element (j-1)

3	4	7
6	2	5
1	3	8

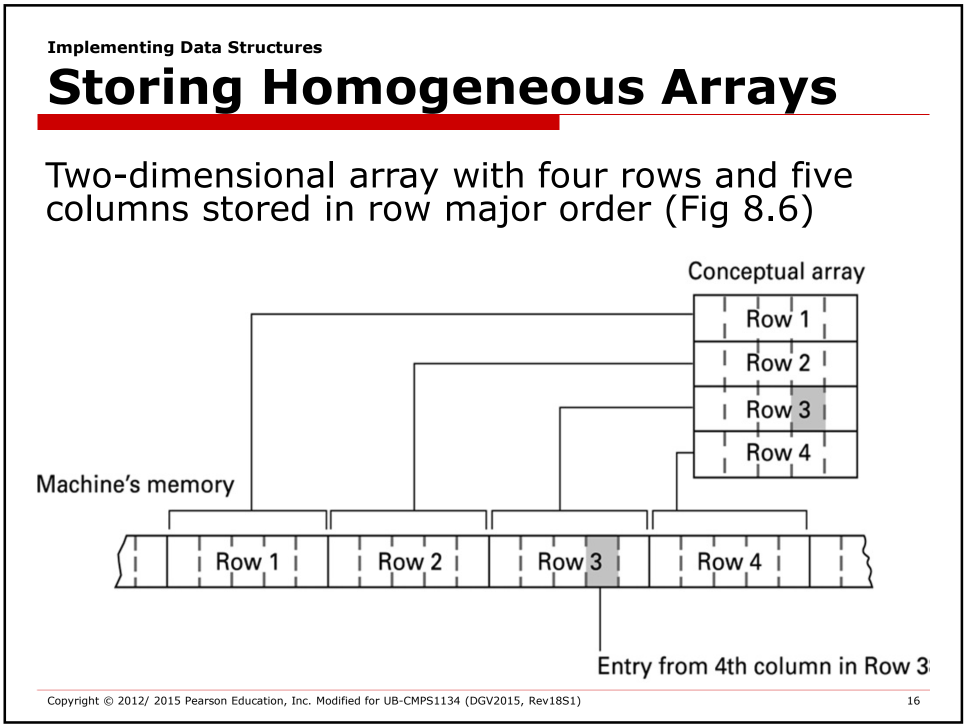




Implementing Data Structures

Storing Homogeneous Arrays

The array of temperature readings stored in memory starting at address x (Fig 8.5)



Implementing Data Structures

Storing Homogeneous Arrays

Two-dimensional array with four rows and five columns stored in row major order (Fig 8.6)

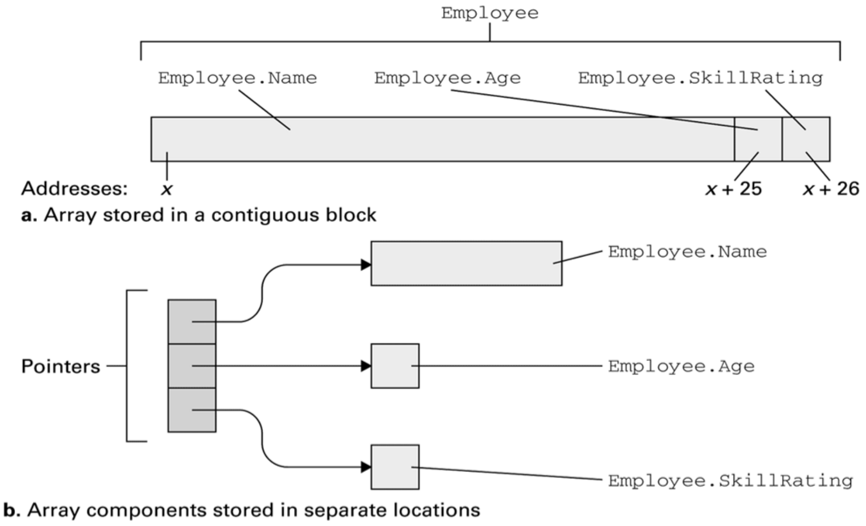
Implementing Data Structures

Storing Heterogeneous Arrays

- □ Fields can be stored one after the other in a contiguous block:
  - Memory cell address of each field can be computed
- □ Fields can be stored in separate locations identified by pointers

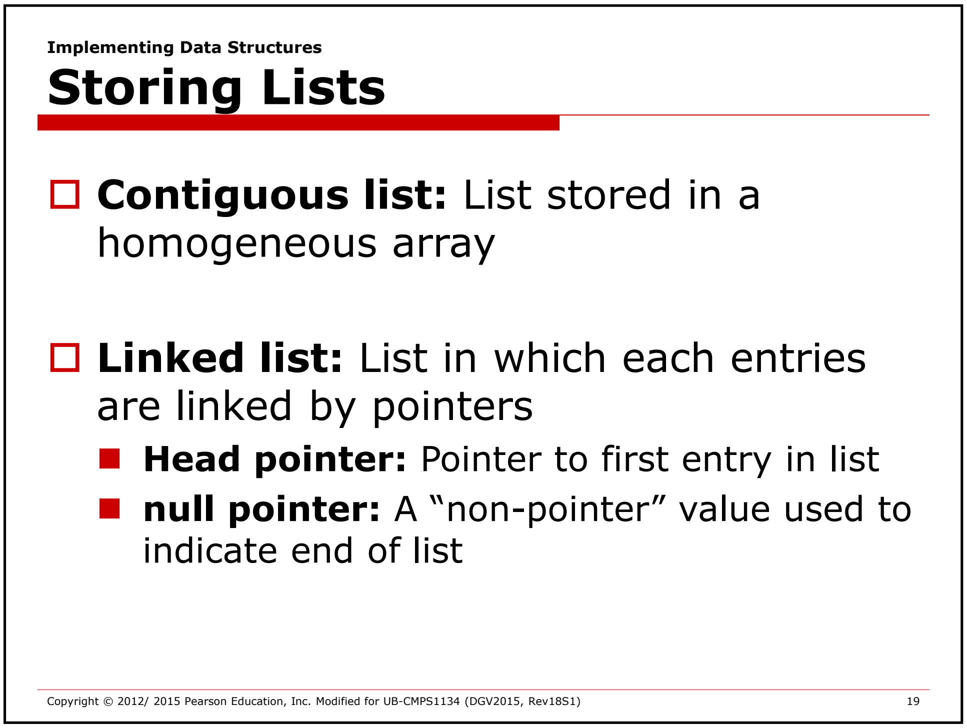
Implementing Data Structures

Storing Heterogeneous Array Employee



a. Array stored in a contiguous block

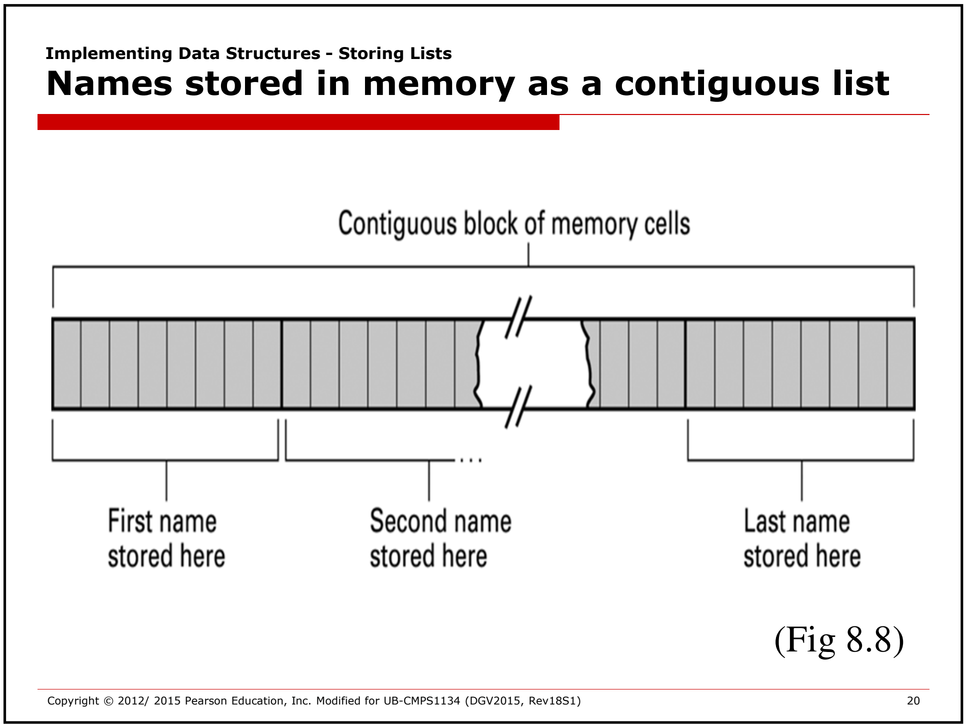
b. Array components stored in separate locations



Implementing Data Structures

Storing Lists

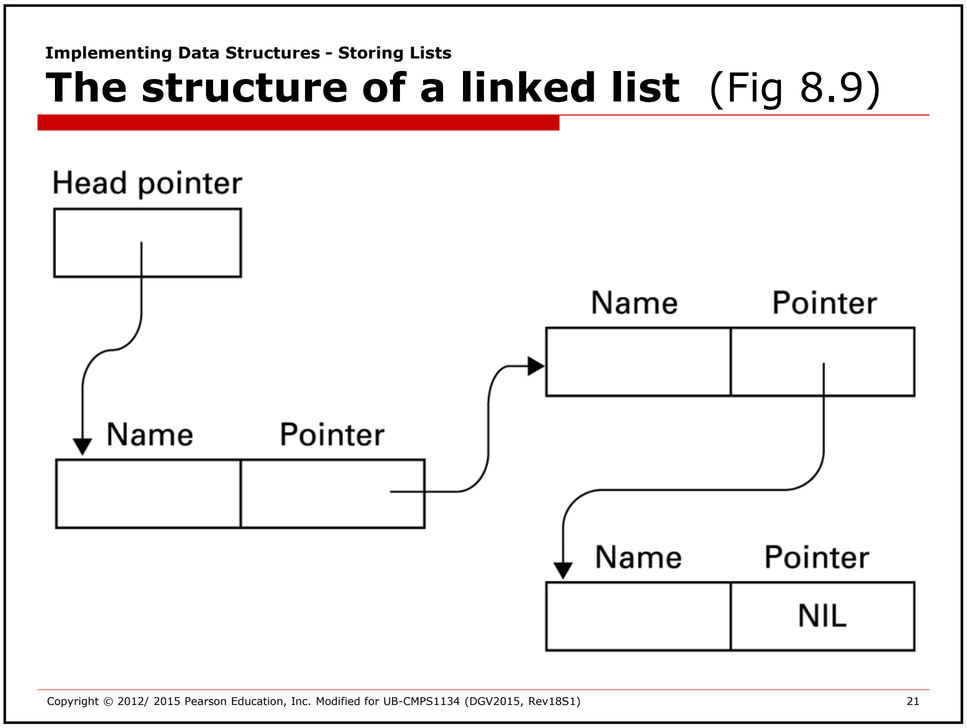
- **Contiguous list:** List stored in a homogeneous array
- **Linked list:** List in which each entries are linked by pointers
  - **Head pointer:** Pointer to first entry in list
  - **null pointer:** A "non-pointer" value used to indicate end of list



Implementing Data Structures - Storing Lists

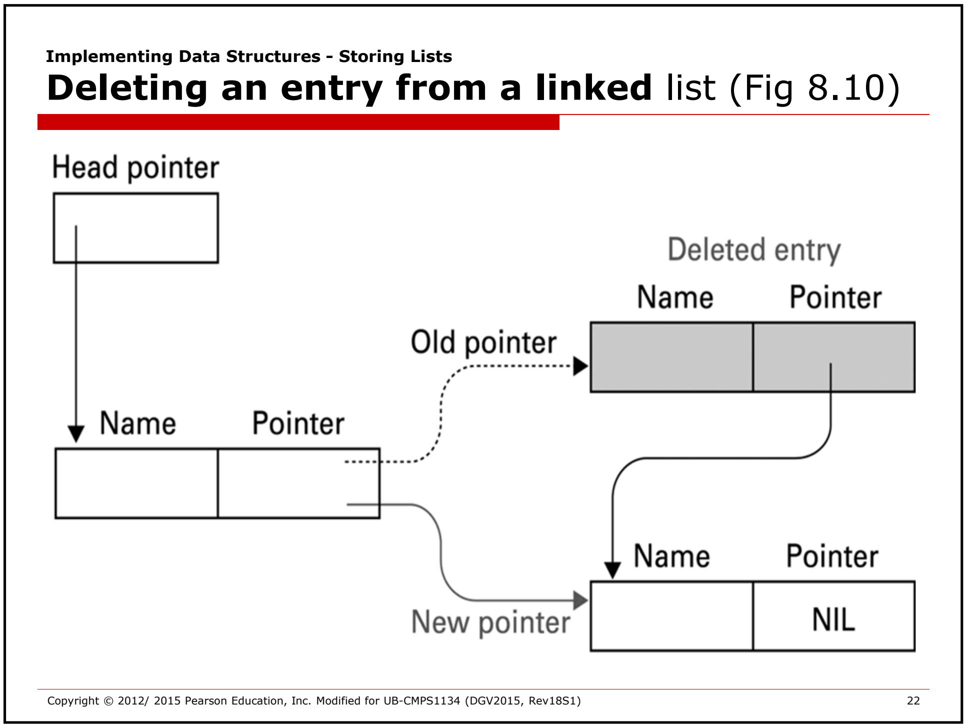
Names stored in memory as a contiguous list

(Fig 8.8)



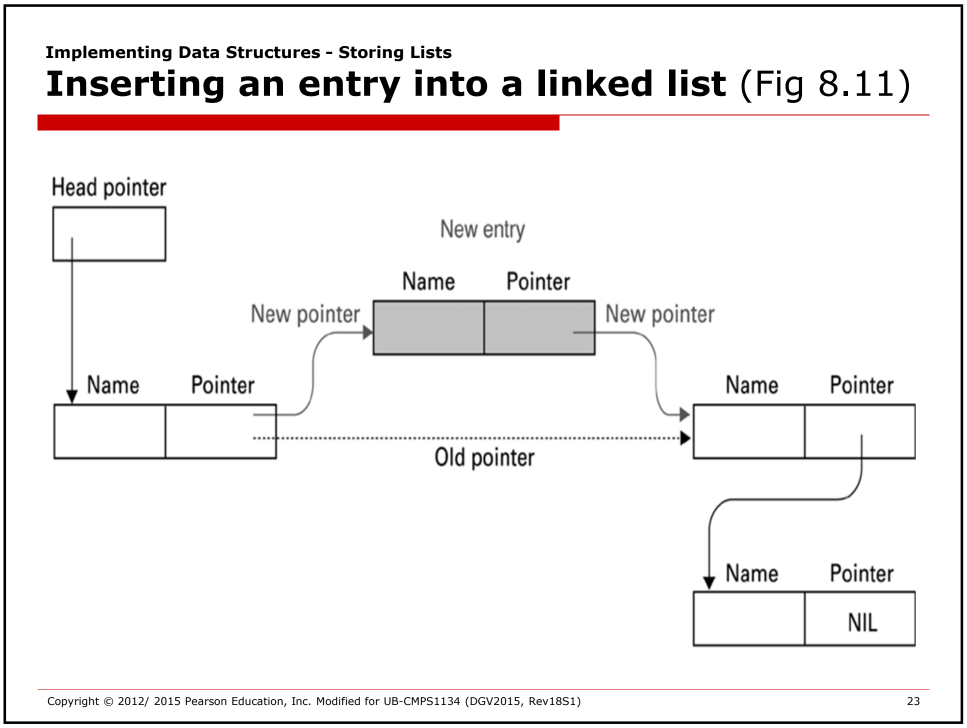
Implementing Data Structures - Storing Lists

The structure of a linked list (Fig 8.9)



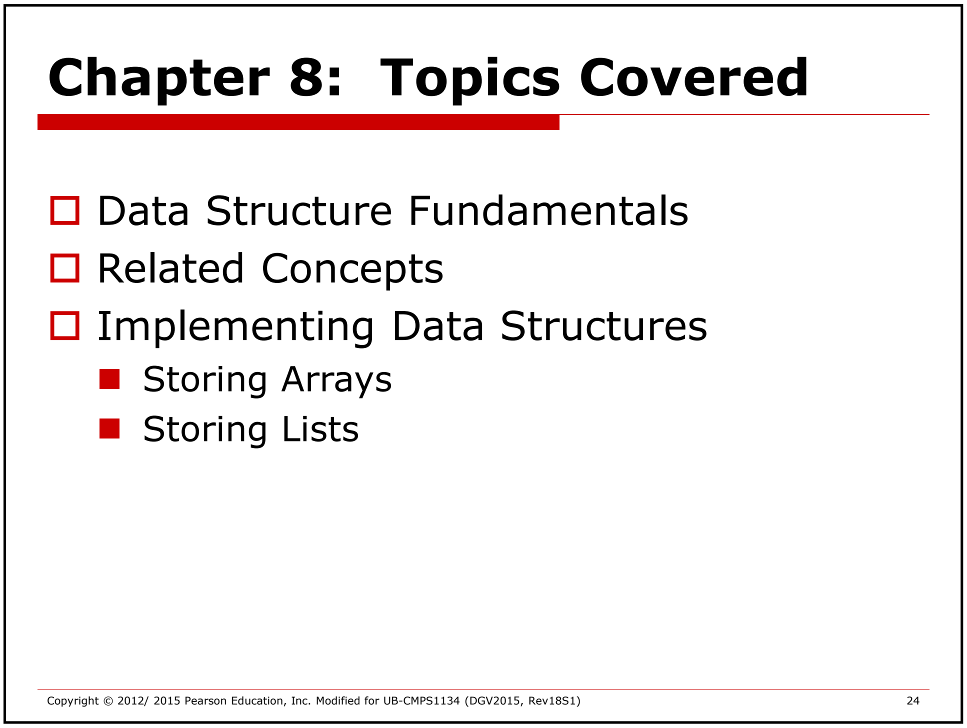
Implementing Data Structures - Storing Lists

Deleting an entry from a linked list (Fig 8.10)



Implementing Data Structures - Storing Lists

Inserting an entry into a linked list (Fig 8.11)



Chapter 8: Topics Covered

- Data Structure Fundamentals
- Related Concepts
- Implementing Data Structures
  - Storing Arrays
  - Storing Lists