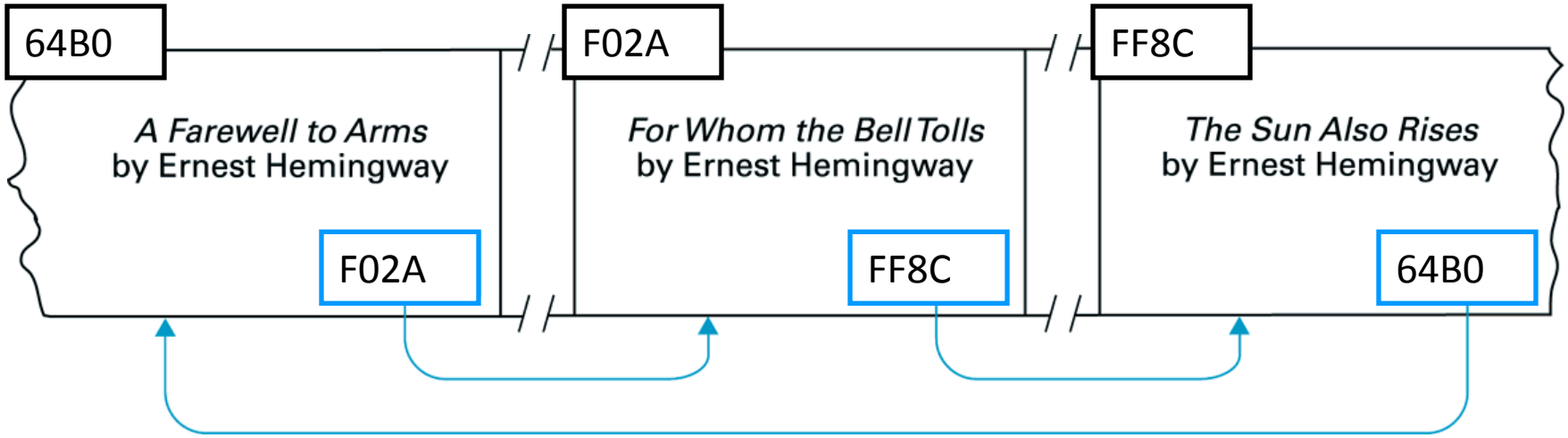
CHAPTER 7

Data Structures

- Abstractions of the actual data organization in main memory
- Allow users to perceive data as ‘logical units’ (e.g.: arrangement in rows and columns)

7.1: Data Structure Basics: Pointers

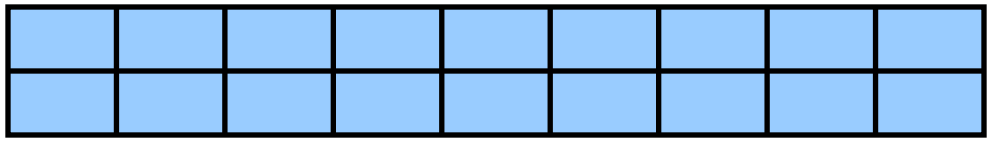
- Pointers:
  - pointer = location in memory that contains the address of another location in memory
  - so: pointer *points* to data positioned elsewhere in memory



7.1: Static versus Dynamic Data Structures

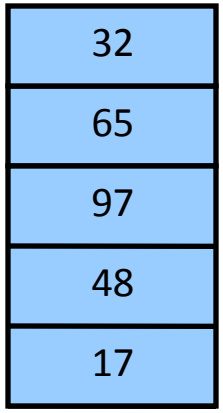
- Static:
  - shape & size of structure does not change over time
    - example in C: `int Table[2][9];`

Table:



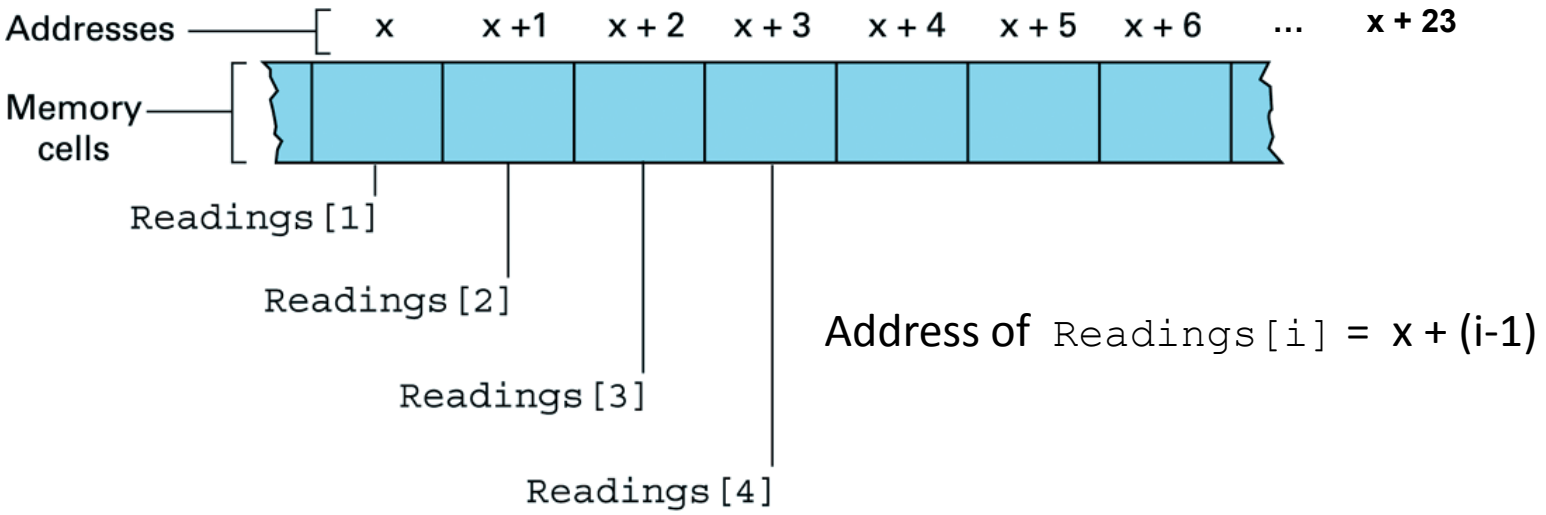
- Dynamic:
  - shape & size may change
  - example: Stack

Stack:



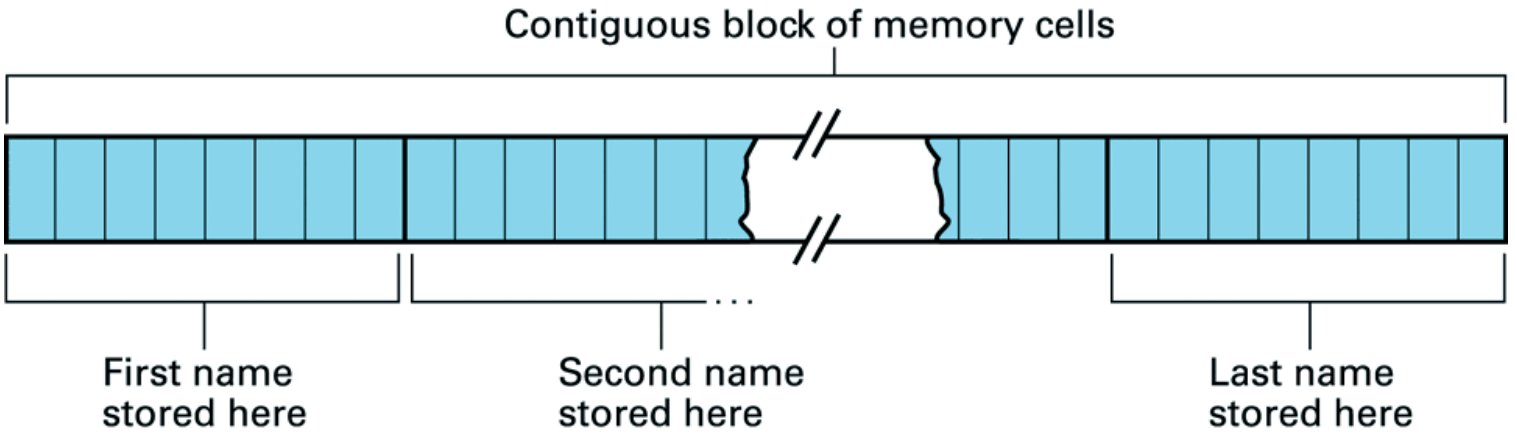
7.2: Arrays

- Example: to store 24 hourly temperature readings…
- … a convenient storage structure is *1-D homogeneous array* of 24 elements (e.g. in C: `float Readings[24]`)
- In main memory:



7.3: Lists

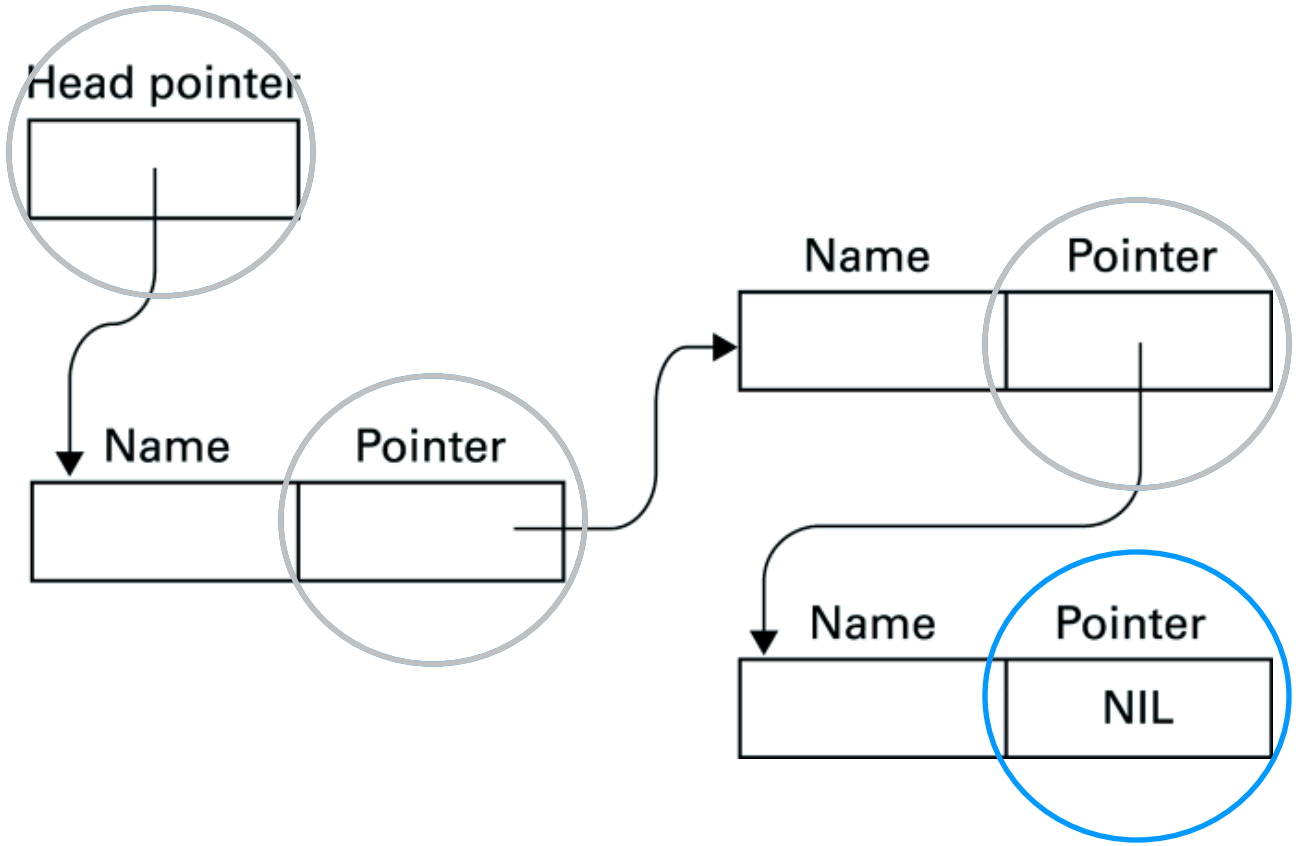
- To store an ordered list of names we could use 2-D homogeneous array (in C: `char Names[10][8]`)



- However:
  - addition & removal of names requires expensive data movements!

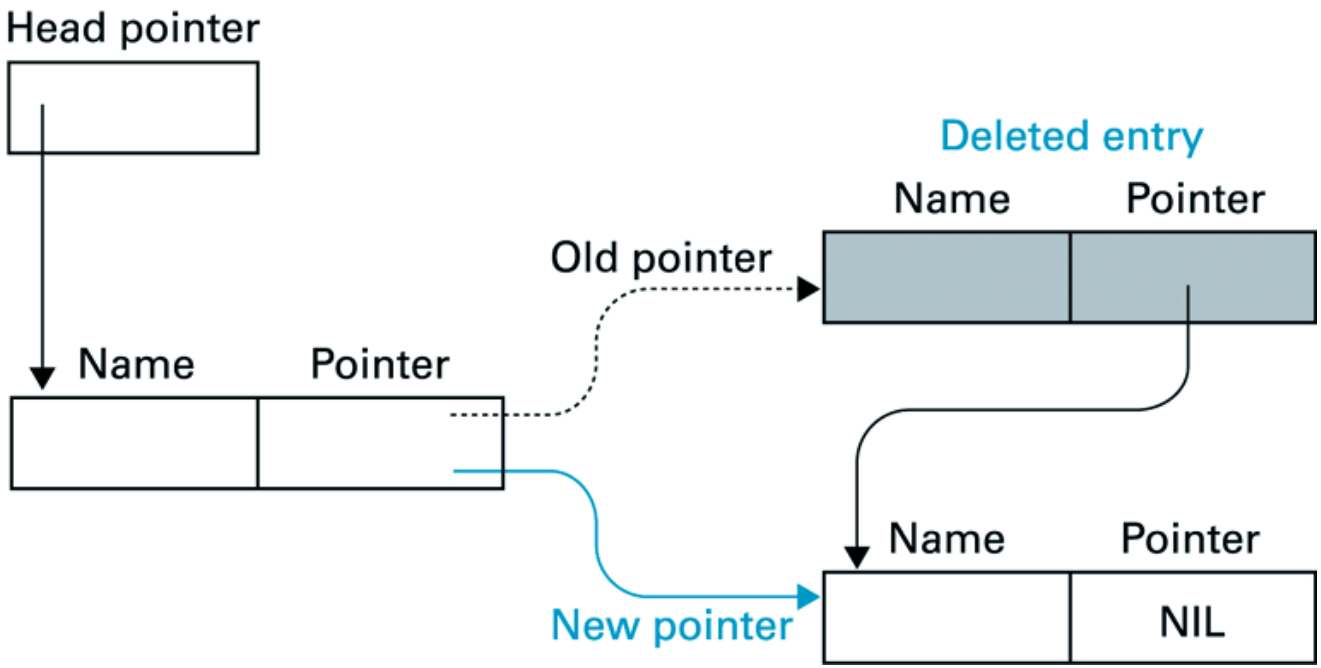
7.3: Linked Lists

- Data movements can be avoided by using a ‘linked list’, including pointers to list entries



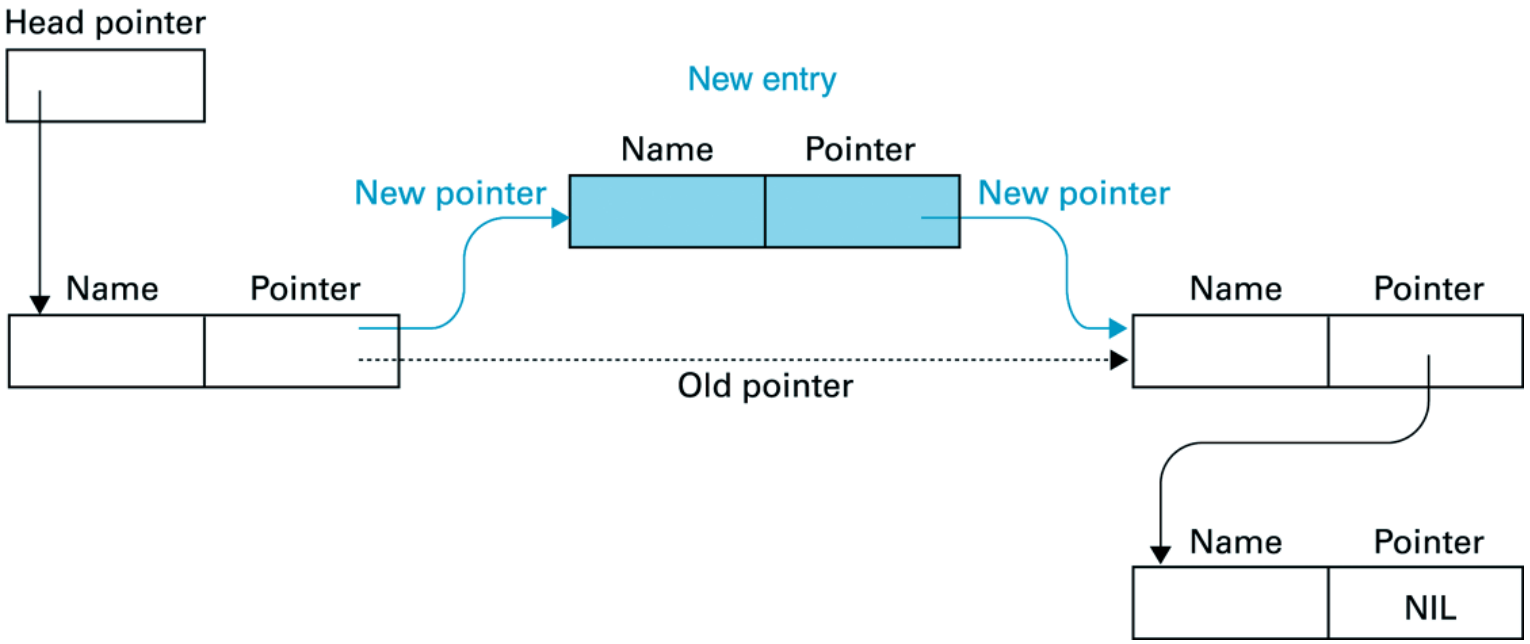
7.3: Deleting an Entry from a Linked List

- A list entry is removed by changing a single pointer:



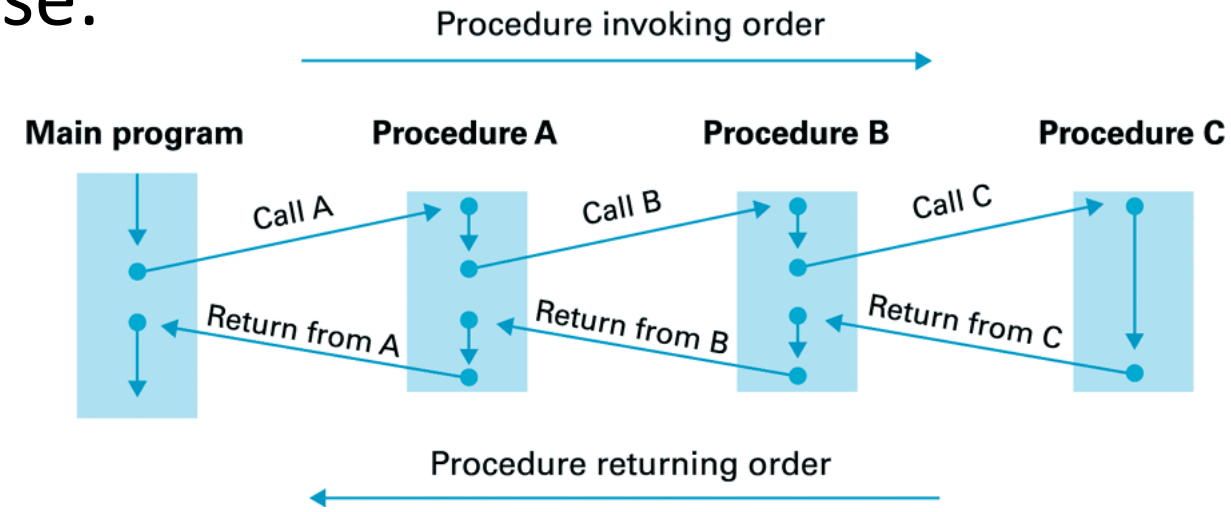
7.3: Inserting an Entry into a Linked List

- A new entry is inserted by setting pointer of
  - (1) new entry to address of entry that is to follow
  - (2) preceding entry to address of new entry:

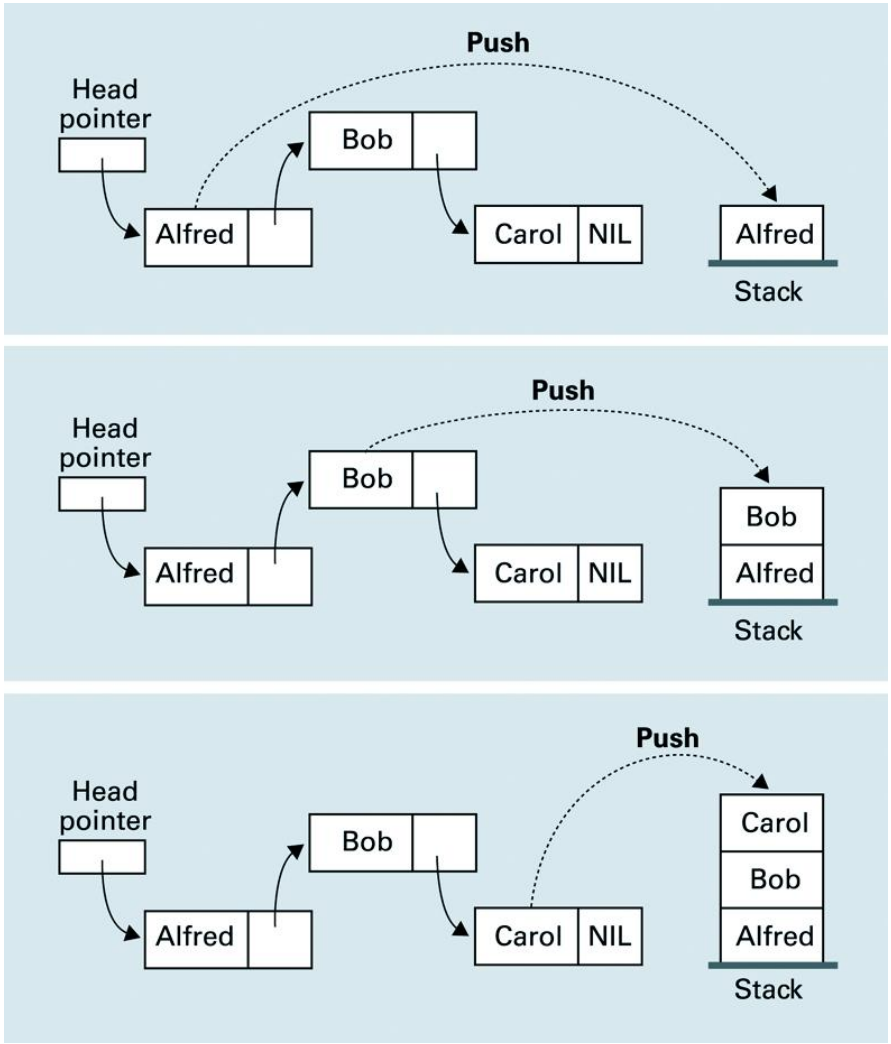


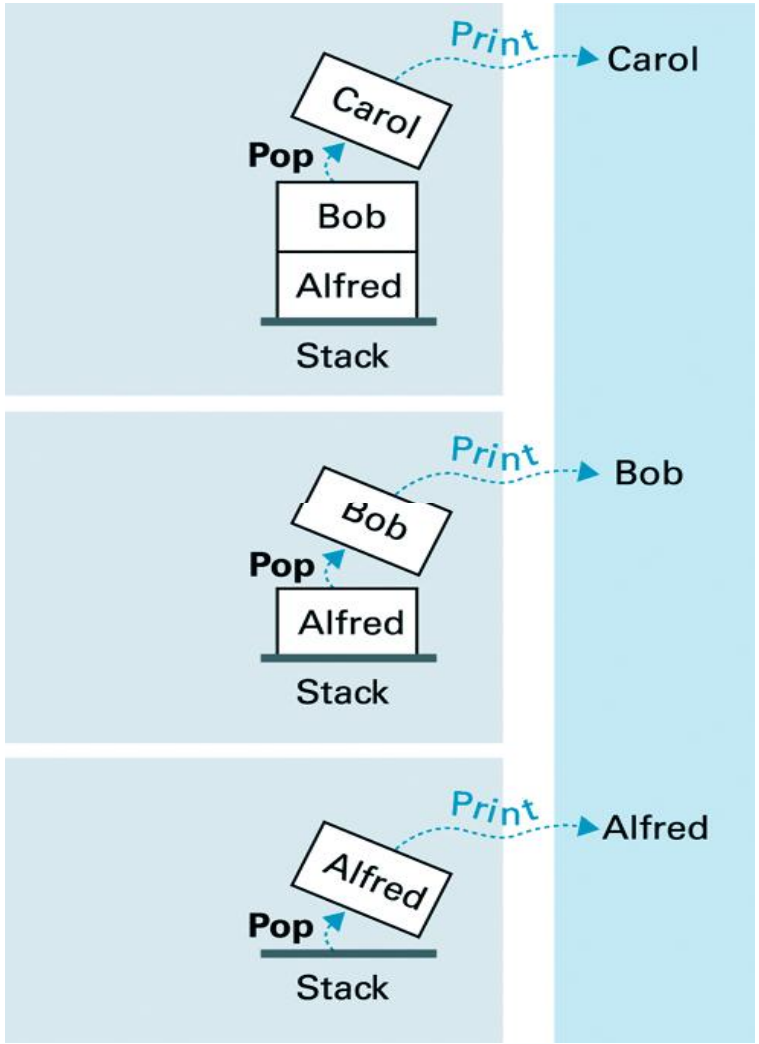
7.4: Stacks

- Disadvantage of contiguous array structures:
  - insertion / removal requires costly data movements
- Still okay if insertion / removal restricted to end of array => stack (with *push* & *pop* operations)
- Typical use:

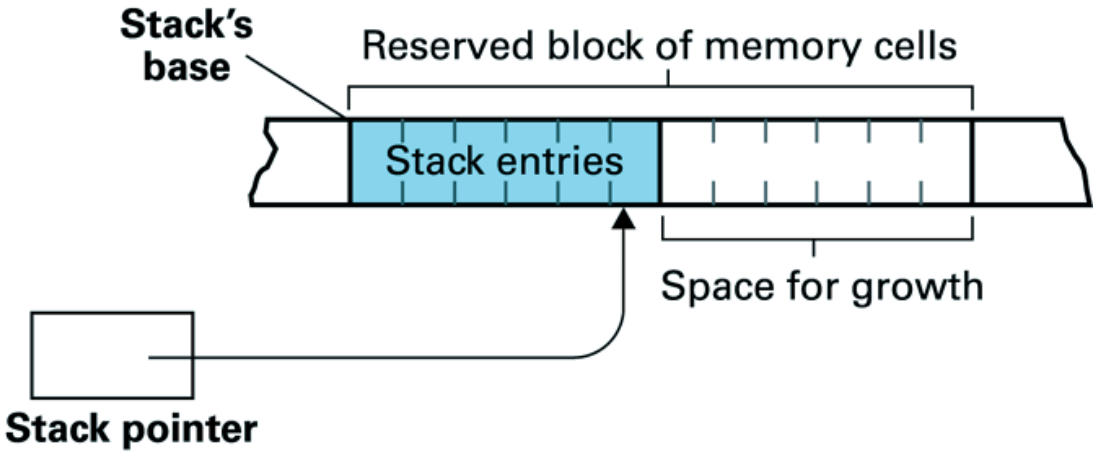


7.4: Push / Pop (to print inverse linked list)





7.4: A Stack in Memory



- Here:
  - conceptual structure close to identical to actual structure in memory
- If maximum stack-size unknown:
  - pointers can be used (=> conceptual = actual structure)

Chapter 7 - Data Structures: Conclusions

- Pointers:
  - basic aid in definition of *dynamic* data structures
- Often used data structures:
  - Arrays
  - Lists
  - Stacks
  - …