Abstract Data Type and Data Structure

Humayun Kabir

Professor, CS, Vancouver Island University, BC, Canada

Abstract Data Type and Data Structure

Outline

- Data Type (DT)
- Data Structure (DS)
- Abstract Data Type (ADT)

Data Type

Specification of Data

- Specifies the kind of data.
- Specifies the size of data, i.e., memory requirement.
- Specifies the range of values.
- Specifies the set of operations on these values.
- Example Data Types
  - Integer, Float, Double, Boolean, Character etc.

Data Type

Example Data Types

Data	Size	Range	Operations
Integer	4 bytes	-2,147,483,648 to 2,147,483,647	+, -, *, /, %
Float	4 bytes	+/- 3.4e +/- 38	+, -, *, /

Atomic and Non-atomic Data

Two kinds of Data

- Atomic Data
- Non-atomic Data

Atomic Data

Atomic Data

- Single unit of data
- Operations must be performed on the whole unit
- Must be of specific type:
  - Integer, Float, Double, Boolean, Character etc.
- Examples: 100, -5, 3.141, 2.718, True, False , A, e, $

Non-atomic Data

Non-atomic Data

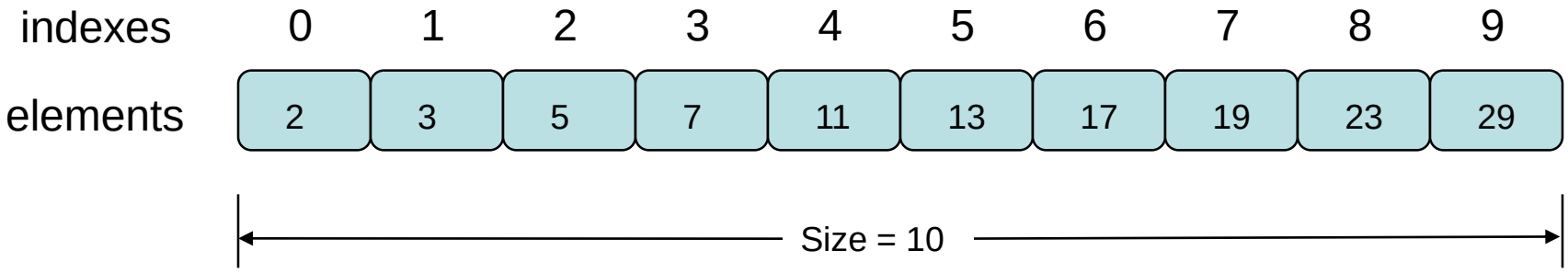
- Composed of more than one atomic data, called elements or fields.
- Has a specific structure to organize these elements/fields.
- Each element can be individually accessed and operated.
- Operations and access sometime depend on the structure.
- Also called User-defined Data Type (UDT).

Data Structure

- The structure of non-atomic or user-defined data is essentially the **Data Structure**

- There are several Data Structures in the literature
    - Array
    - Linked List
    - Record
    - Tree
    - Heap
    - Graph

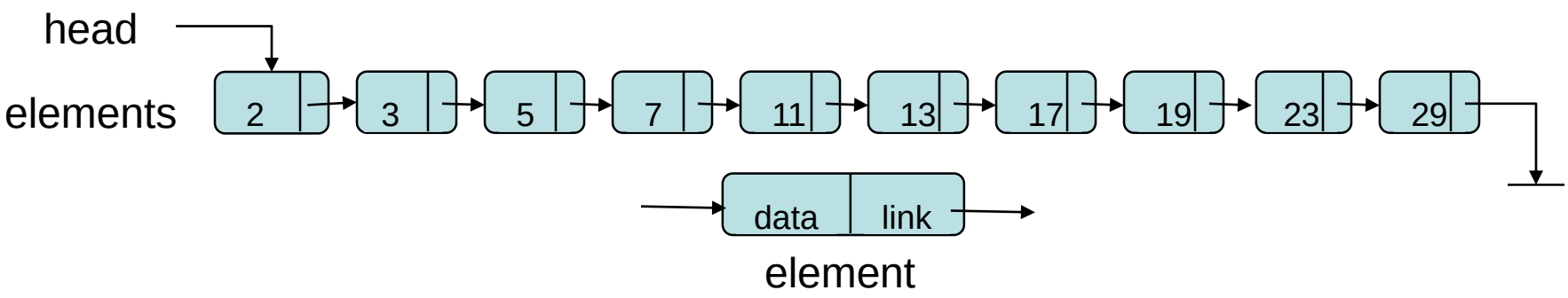
Array Data Structure



Container to hold some elements of same data type

- Has a fixed size.
- Elements are indexed and stored in consecutive memory locations.
- Accessed by index.
- Operations: Insert, Delete, Update, Search, Traverse

Linked List Data Structure



A data structure where elements are linked

- Each element has a data and link.
- Elements are stored in arbitrary memory locations.
- Size of a linked list is not fixed
- Accessed by link.
- Operations: Insert, Delete, Update, Search, Traverse

Abstract Data Type

Abstract Specification of Data Type

- Separates the notions of **specification** and **implementation** of a data type.

- The separation is achieved by
  - specifying a data type only by its operations.
  - not specifying underlying data structure and algorithms to implement these operations, i.e., no implementation details.

Abstract Data Type

- Any specific implementation of an ADT is called a Concrete Data Type (CDT)
  - A CDT has to implement all the operations of its ADT as specified.
  - A CDT is free to choose the data structure and the algorithms to implement the operations.
  - A CDT can be easily replaced by another CDT if both implement the same ADT.

Abstract Data Type

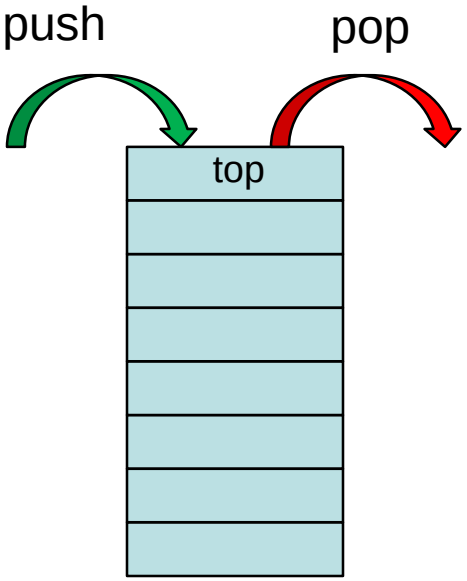
ADT Examples

- Stack
- Queue
- Priority Queue
- List
- Set
- Map

Abstract Data Type: Stack

Stack in real life: Stack of Books

Source:
https://www.freepik.com/free-photo/stack-of-various-books-on-a-table_991451.htm



Last in First out (LIFO)

Abstract Data Type: Stack

Stack ADT Specification

- push()
  - Adds an element on top of stack
- pop()
  - Removes an element from the top of the stack
- peek()
  - Examines the top element without removing it
- getSize()
  - Returns the number of elements in the stack
- isEmpty()
  - Checks whether the stack has any element in it

Stack CDT with Array

Stack CDT with Linked List

Abstract Data Type

ADT Benefits

- ADT allows different implementations from the same abstraction, i.e., adds more flexibility.
- ADT allows seamless switch from one implementation to another, i.e., makes it more maintainable.
- ADT eases to make the code modular, i.e., manageable.
- ADT facilitates software testing.

Conclusion

Learned

- Abstract Data Type (ADT)
  - Stack ADT
  - Stack CDT with Array
  - Stack CDT with Linked List

Home Work

Home Work

- Queue (FIFO)
  - Queue ADT
  - Queue CDT with Array
  - Queue CDT with Linked List

Home Work

Queue ADT Specification

- enqueue()
  - Adds an element at the end of a queue.
- dequeue()
  - Removes an element from the front of the queue.
- peek()
  - Examines the front element without removing it
- getSize()
  - Returns the number of elements in the queue.
- isEmpty()
  - Checks whether the queue has any element in it