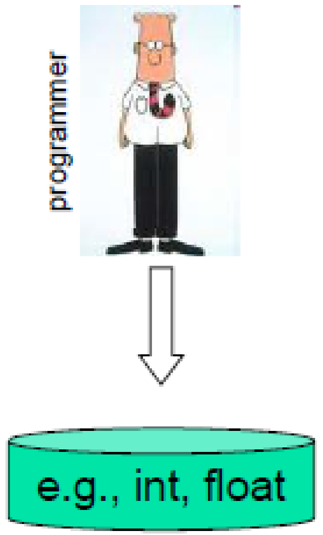
DATA STRUCTURE & ALGORITHMS

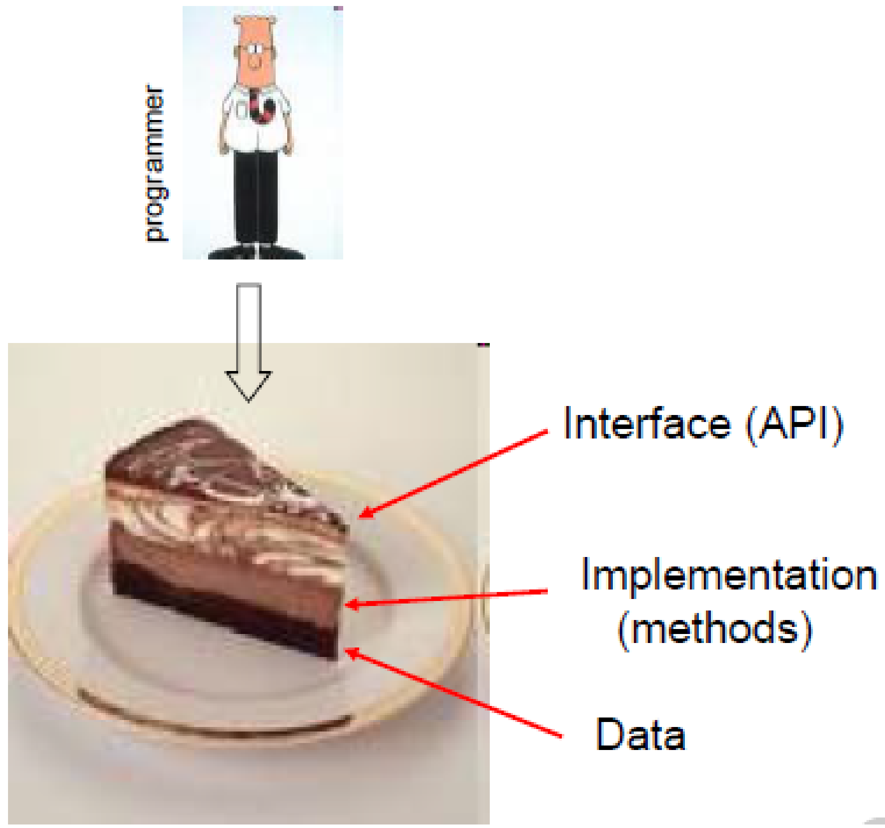
LECTURE - 2

Primitive Data Type vs. Abstract Data Types

Primitive DT:



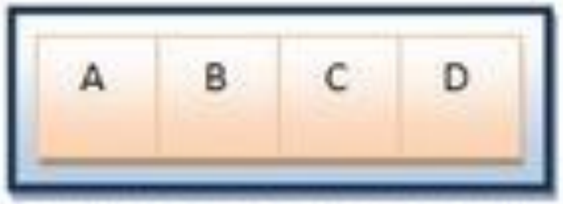
ADT:



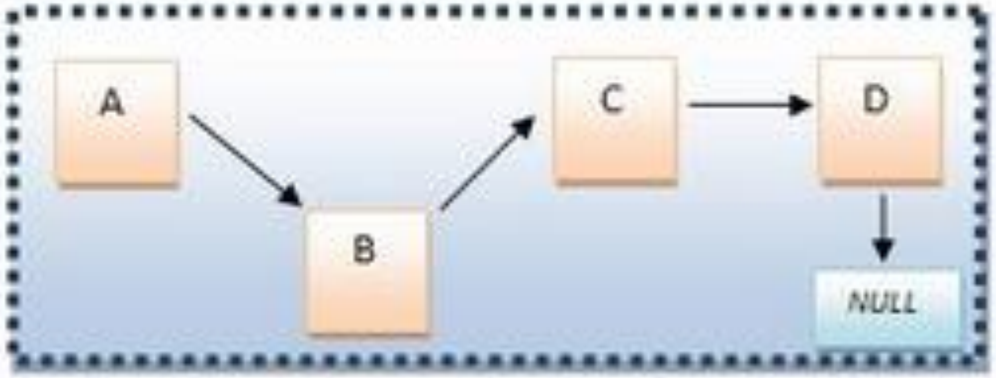
ABSTRACT DATA TYPES

- An Abstract data type (ADT) refers to a set of data values and associated operations that are specified accurately, independent of any particular implementation.
- With an ADT, we know what a specific data type can do, but how it actually does it is hidden.
- **Here are some examples.**
- Stack: operations are "push an item onto the stack", "pop an item from the stack", "ask if the stack is empty"; implementation may be as array or linked list or whatever.
- Queue: operations are "add to the end of the queue", "delete from the beginning of the queue", "ask if the queue is empty"; implementation may be as array or linked list or heap.
- Search structure: operations are "insert an item", "ask if an item is in the structure", and "delete an item"; implementation may be as array, linked list, tree, hash table, ...

STATIC AND DYNAMIC IMPLEMENTATION OF DATA STRUCTURE



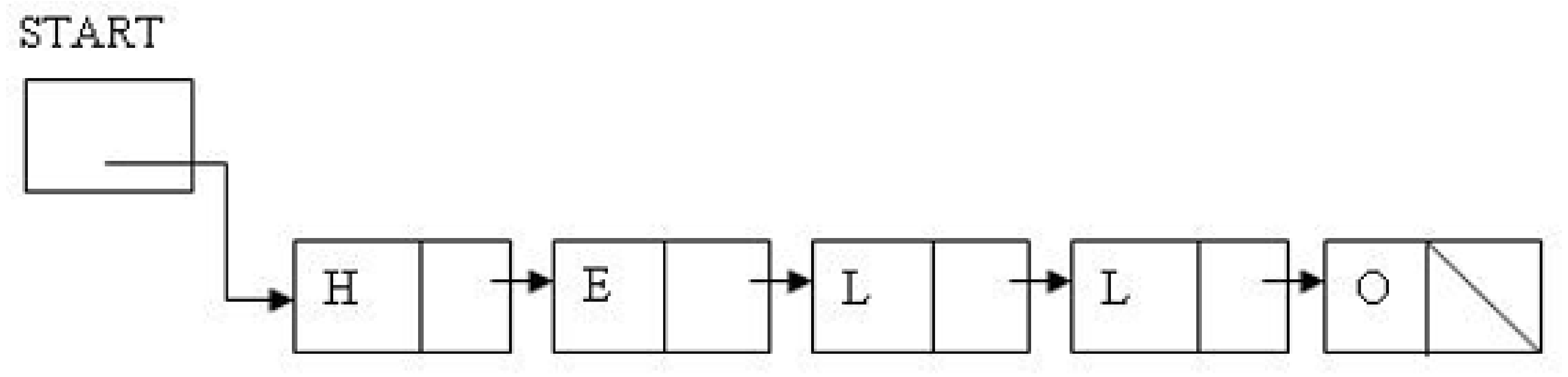
Array (static structure) of four elements.



Linked list (dynamic structure) of four elements. Each element pointing to the next element

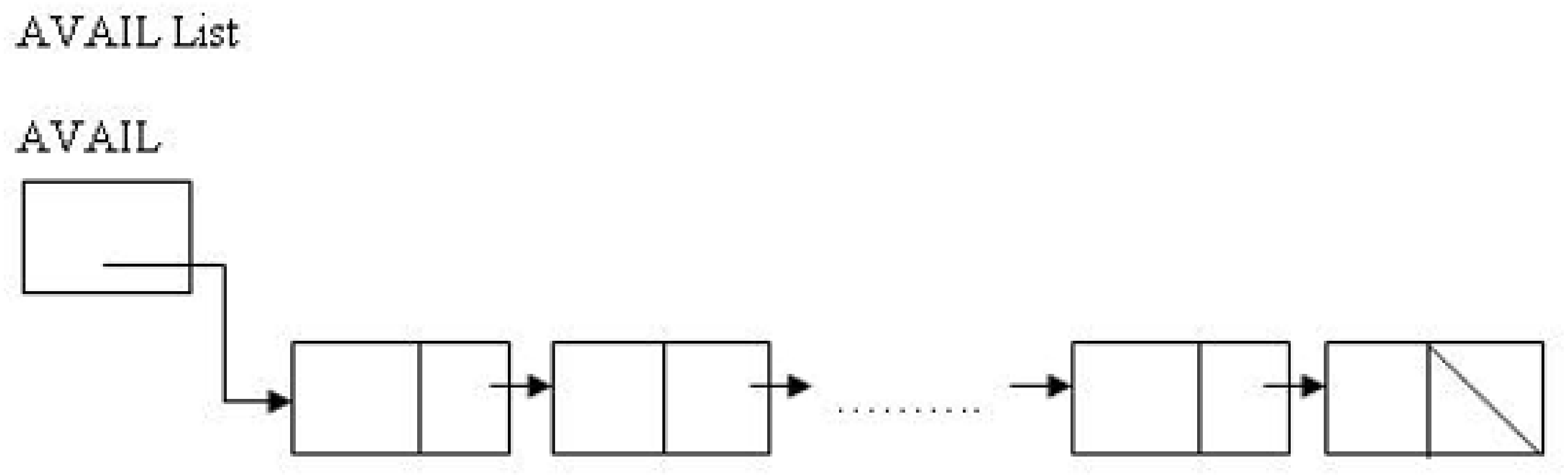
LINKED LIST

- A linked list or one-way list is a linear collection of data elements, called nodes, where the linear order is given by means of pointers.



MEMORY ALLOCATION: GARBAGE COLLECTION

- Together with the linked lists in memory , a special list is maintained which consists of unused memory cells. This list, which has its own pointer, is called list of available space or the free storage list or the free pool.
- This free storage list will also be called the AVAIL list.



OVERFLOW AND UNDERFLOW CONDITIONS

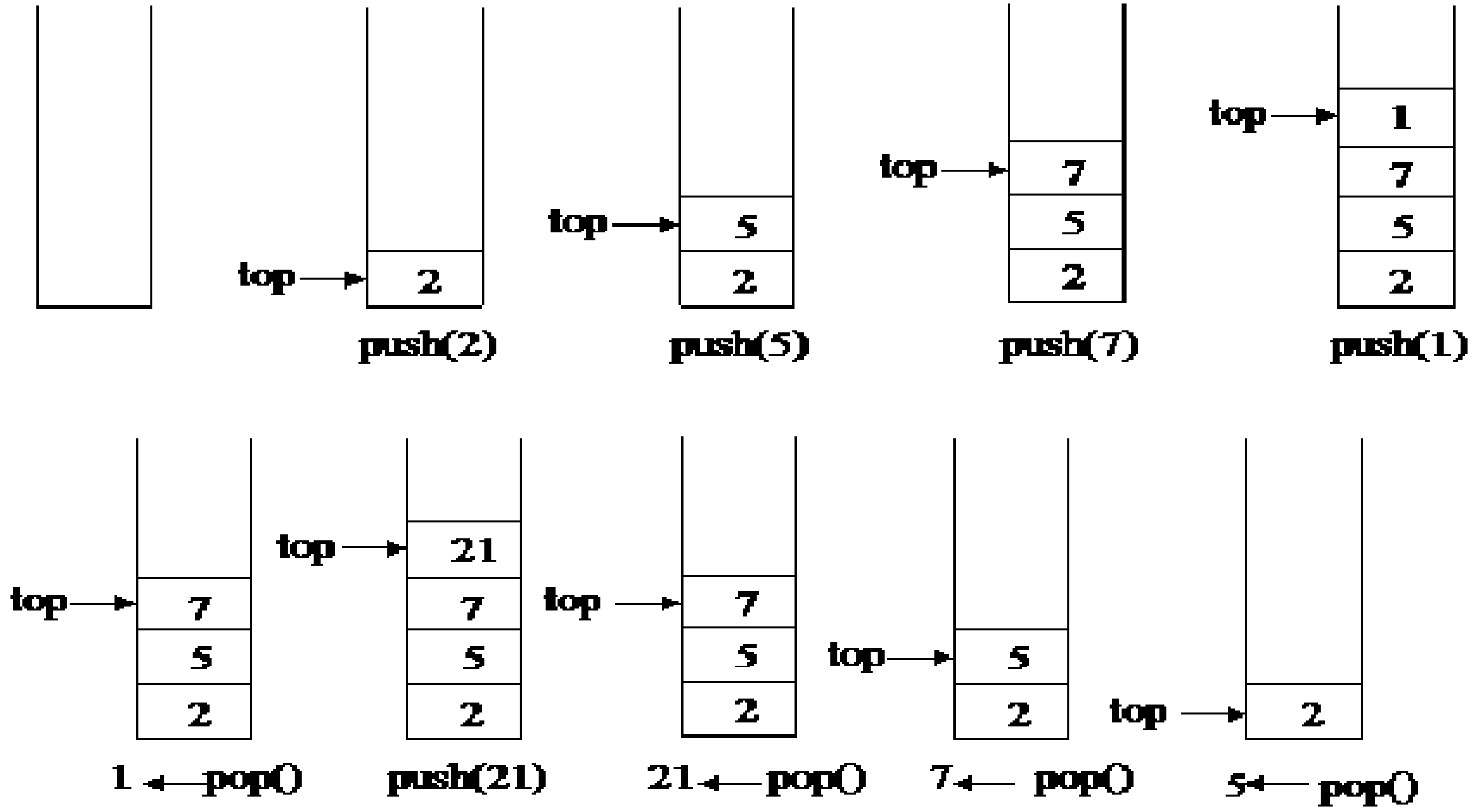
- Overflow will occur with linked lists when AVAIL = NULL and there is an insertion.

- Underflow will occur with linked lists when START = NULL and there is a deletion.

STACK

A stack is a list of elements in which an element may be inserted or deleted only at one end, called the top of the stack.

Special terminology is used for two basic operations associated with stacks:
"Push" is the term used to insert an element into a stack.
"Pop" is the term used to delete an element from a stack.



STACK

Values of Stack and Top :

Operation	Explanation
top = -1	-1 indicated Empty Stack
top = top + 1	After push operation value of top is incremented by integer 1
top = top – 1	After pop operation value of top is decremented by 1

ARRAY REPRESENTATION OF STACKS

Stack may be represented in the computer in various ways, usually by means of a one-way list or a linear array..

Unless otherwise stated, each of our stacks will be maintained by

- **A linear array STACK**
- **A pointer variable TOP**
- **A variable MAXSTK**

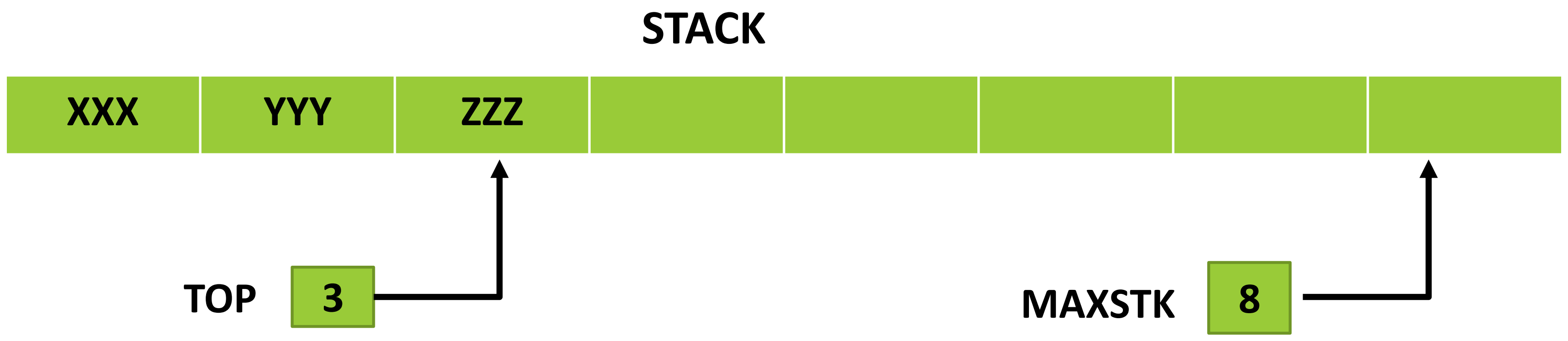


Fig: Array Representation of Stack

ALGORITHM FOR INSERTION (PUSH OPERATION)

PUSH(STACK, TOP, MAXSTK, ITEM)

This procedure pushes an ITEM onto a stack.

```
1.  [Stack already filled]
     if TOP = MAXSTK, then Print: OVERFLOW, and return.
2. Set TOP := TOP+1. [ Increase TOP by 1].
3. Set STACK[TOP] := ITEM. [Inserts ITEM in new TOP position.].
4. Return.
```

ALGORITHM FOR DELETION (POP OPERATION)

POP(STACK, TOP, ITEM)

This procedure deletes the top element of the STACK and assign it to the variable ITEM.

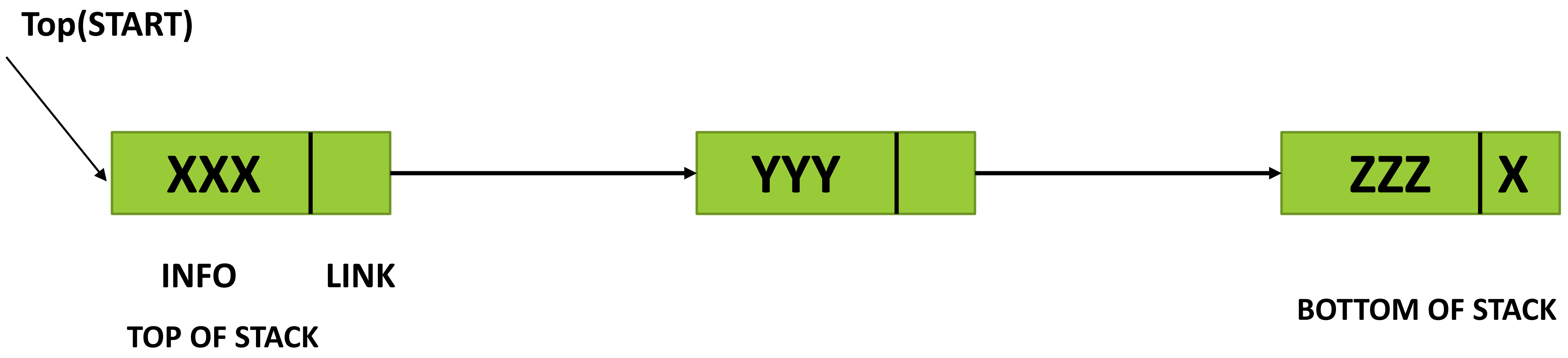
```
1.  [Stack has an item to be removed].
    If TOP = 0, then: Print: UNDERFLOW, and Return.
2. Set ITEM:= STACK[TOP]. [Assign TOP element to ITEM].
3. Set TOP:= TOP – 1.[Decreases TOP by 1].
4. Return.
```

MINIMIZING OVERFLOW

- The number of elements in a stack fluctuates as elements are added or removed from a stack.
- Accordingly, the particular choice of the amount of memory for a given stack involves a time- space tradeoff.

LINKED REPRESENTATION OF STACKS

The linked representation of a stack, commonly termed linked stack is a stack that is implemented using a singly linked list.



ALGORITHM FOR INSERTION

PUSH_LINKSTACK(INFO, LINK, TOP, AVAIL, ITME)

This procedure pushes an ITEM into a linked stack.

```
1.[Available space?] If AVAIL = NULL, the write OVERFLOW and Exit.

2.[Remove first node from AVAIL List].

    Set NEW :=AVAIL and AVAIL :=LINK[AVAIL].

3. Set INFO[NEW] := ITEM [Copies ITEM into new node].

4. Set LINK[NEW] := TOP [ New node points to the original top node in the stack].

5. Set TOP = NEW [ Reset TOP to point to the new node at the top of the stack].

6. Exit.
```

ALGORITHM FOR DELETION

POP_LINKSTACK(INFO, LINK, TOP, AVAIL, ITME)

This procedure deletes the top element of a linked stack and assigns it to the variable ITEM.

```
1.[Stack has an item to be removed?]
    If TOP = NULL, the write UNDERFLOW and Exit.
2. Set ITEM:= INFO[TOP]. [ Copies the top element of stack into ITEM].
3. Set TEMP := TOP and TOP = LINK[TOP].
4. [Return deleted node to the AVAIL List].
    Set LINK[TEMP] = AVAIL and AVAIL = TEMP.
5. Exit.
```