Second stage : Data Structures

Data Structure.- Stack

israa Mahmoud hayder

Non-Primitive Data Structure
Linear Data Structure: Stack

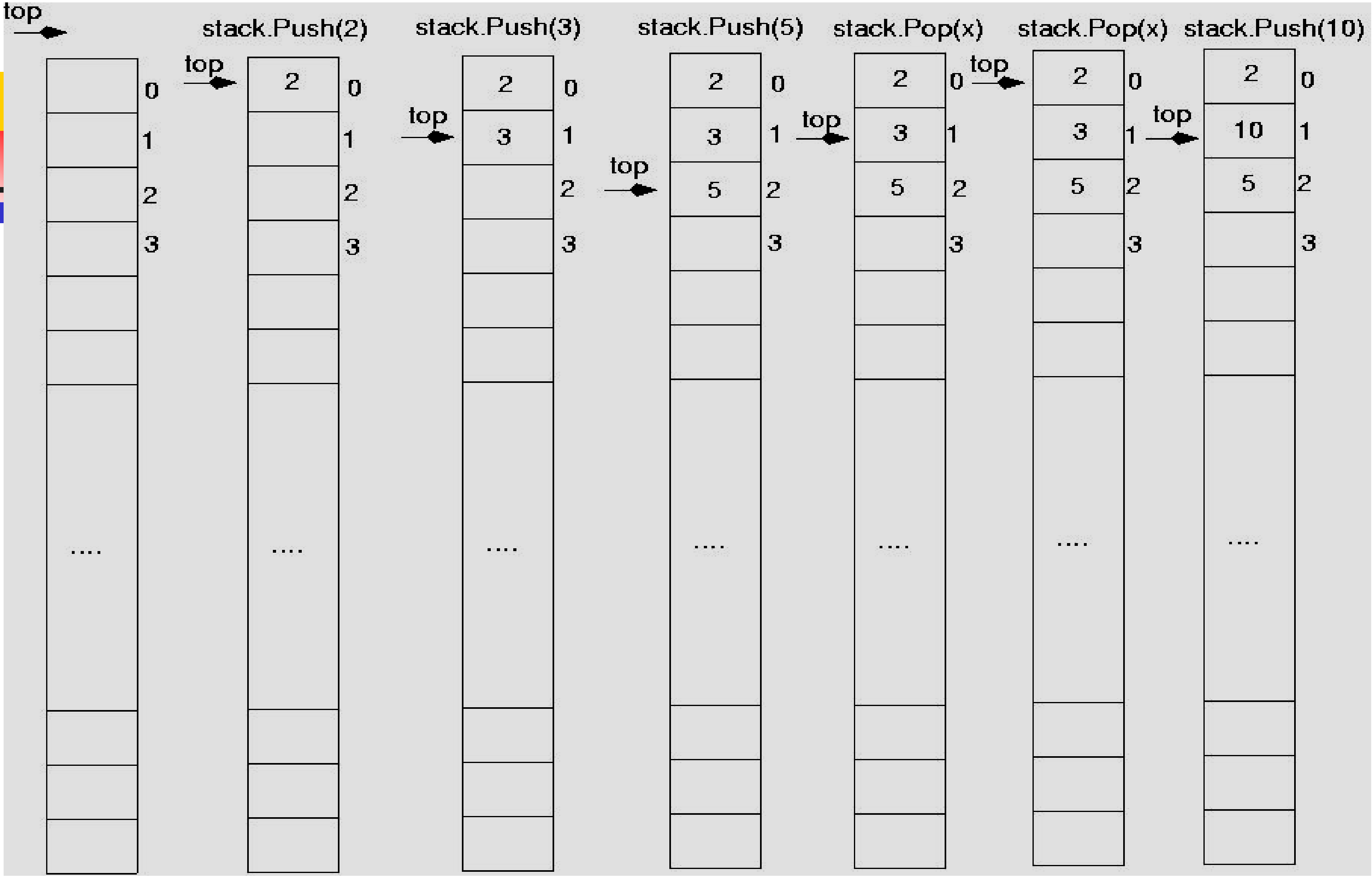
- • A stack is a linear data structure which can be accessed only at one of its
- ends for storing and retrieving data. **For this reason, a stack is called an**
- **LIFO structure: last in/first out**.
- 2
- **F**
- **E**
- **D**
- **C**
- **B**
- **A**
- So **F** is the current top element of the stack, If
- any new items are added to the stack they are
- placed on top of **F**, and if any items are
- deleted, **F** is the first to be deleted.

Stack Specification

- Definitions: (provided by the user)
    - MAX_ITEMS: Max number of items that might be on the stack
    - ItemType: Data type of the items on the stack
- Operations
    - MakeEmpty
    - Boolean IsEmpty
    - Boolean IsFull
    - Push (ItemType newItem)
    - Pop (ItemType& item)

Push (ItemType newItem)

- *Function*: Adds newItem to the top of the stack.
- *Preconditions*: Stack has been initialized and is not full.
- *Postconditions*: newItem is at the top of the stack.



stack.Push(2) stack.Push(3) stack.Push(5) stack.Pop(x) stack.Pop(x) stack.Push(10)

Stack overflow

- The condition resulting from trying to push an element onto a full stack.

```
if(!stack.IsFull())
  stack.Push(item);
```

Stack underflow

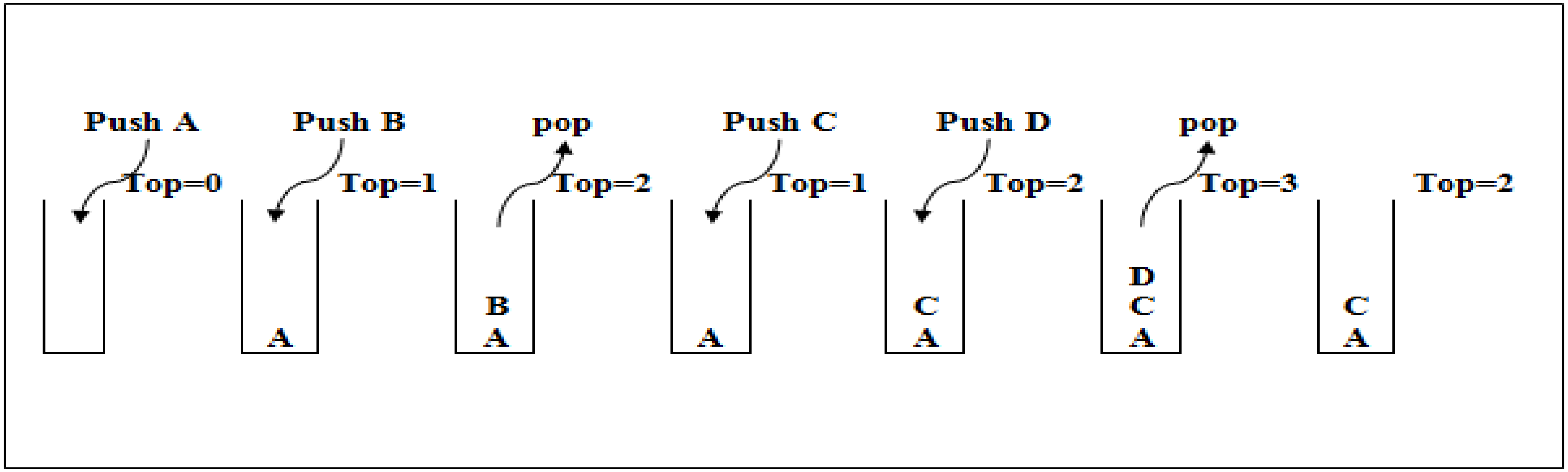
- The condition resulting from trying to pop an empty stack.

```
if(!stack.IsEmpty())
  stack.Pop(item);
```

- **Stack: Application**
- 1. Internet Web browsers store the addresses of recently visited sites on a
- stack. Each time a user visits a new site, that site’s address is “pushed”
- onto the stack of addresses. The browser then allows the user to “pop”
- back to previously visited sites using the “back” button.
- 2. Text editors usually provide an “undo” mechanism that cancels recent
- editing operations and reverts to former states of a document. This undo
- operation can be accomplished by keeping text changes in a stack.

Stack Operations

- The two main operations which can be applied to a stack are given special names, when an item is added to a stack, it is **Pushed** to the stack, and when an item is removed, it is **Popped** from the stack.



- **Stack Operations**
- **These are two basic operations associated with stack:**
- **1. Push():** Insert element e at the top of the stack.
- **2. Pop():** Remove the top element from the stack; an error occurs if the stack
- is empty.
- **Additionally, these supporting functions:**
- **1. size():** Return the number of elements in the stack.
- **2. Isempty():** Return true if the stack is empty and false otherwise.
- **3. Isfull():** Return true if the stack is full and false otherwise.

- **Representation of Stack**
- • Since a stack is a linear data structure, any linear data structure
- implementation will do. A stack can be implemented by means of Array,
- Structure, Pointer, and Linked List. Stack can either be a fixed size one or it
- may have a sense of dynamic resizing
- **1. Non-linked- structures (The array).**
- **2. Linked structures (Linked list).**

- **Stack Representation: Array**
- • The simplest method to represent a stack is to use an array to be home of
- the stack.
- • The stack may therefore be declared and containing two objects: an **array**
- with suitable size and with suitable data type (Int, Float,..etc) to hold the
- elements of the stack, and an **integer** to indicate the position of the current
- stack top within the array.
- **Ex: The below declaration example in the C++ language**
- **const SIZE= 10;**
- **Int stack[SIZE];**
- **Int top= -1; //** That is mean the stack empty.

- **Stack Representation: Array**
- **Sub program to empty the stack**
- **void clearstack ()**
- **}**
- **top = -1 ;**
- **{**
- **Sub program to sure if the stack is full or not**
- **int fullstack()**
- **}**
- **if(top>=size-1)**
- **return(1);**
- **else return(0);**
- **{**

Stack Representation: Array

Sub program to delete an element from the stack

```
void pop()
}
if(emptystack())
}
cout<<"error…the stack is empty"<<endl;
cout<<"press any key to exit"<<endl;
getch();
exit(0);
{
Else {
item=stack[top];
top=top-1;
{{
```