Zaman University

Data Structures and Algorithms

Chapter 2

Abstract Data Types

Outline

- Stacks
- Queues and Priority Queues
- Linked Lists
- Abstract Data Types
- Specialized Lists

Outline

- Stacks
- Queues and Priority Queues
- Linked Lists
- Abstract Data Types
- Specialized Lists

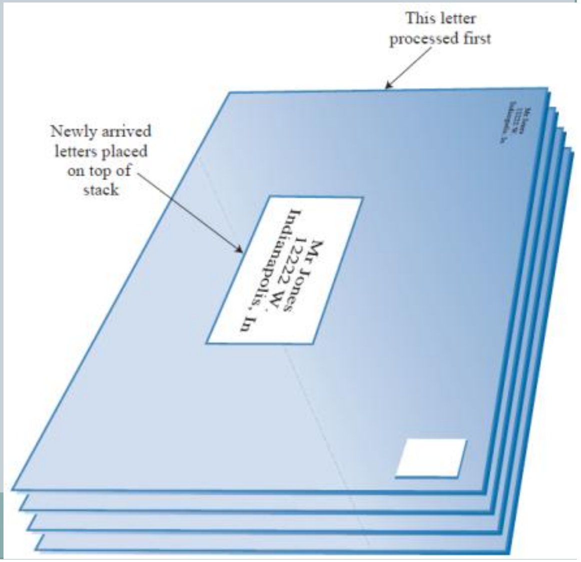
Stacks: Stacks Differ Philosophically from Arrays

- An array – any item can be accessed, if its index number is known;
- In Stacks and Queues, access is restrict – only one item can be read or removed at a given time;
- Thus, interface of these structures is designed to enforce this restrict access. Means that access to other items is (in theory) not allowed.

Understanding Stacks[1]

- The first one provide by the U. S. Postal Service
- Many people, when they get mail, toss it on a stack on the hall table
- They process the accumulated mail from the top down
- First they open the letter on the top and take appropriate action (paying bill) and throw it away
- The first letter have been disposed of, they examine the next letter down, which is now the top of stack
- Eventually, they work their way down to the letter on the bottom of the stack

- Sure, many people do NOT follow this top-to-bottom approach
- They may take the letter of bottom first, to process the oldest first
- In this case, their mail system is no longer a stack, if they take off letters of the bottom, it is a queue


This letter processed first
Newly arrived letters placed on top of stack
Mr Jones
12222 W.
Indianapolis, In

Understanding Stacks[2]

- Another stack analogy is the tasks you perform during a typical workday
- You are busy with a long term project A
- But you are interrupted by co-worker asking you for temporary help with another project B
- While you are working on project B, someone from accounting stops by for a meeting about travel expenses (C)
- During this meeting, you get an emergency call from someone in Sale and spend few minutes troubleshooting a product (D)

- When you finish calling D, you resume meeting C, when you finish with C, you resume project B, when you finish with B, you can finally get back to project A.

Stacks: New Term

- Placing a data item on the top of the stack is called ***pushing***
- Removing it from the top of the stack is called ***popping*** it
- These are the primary stack operations

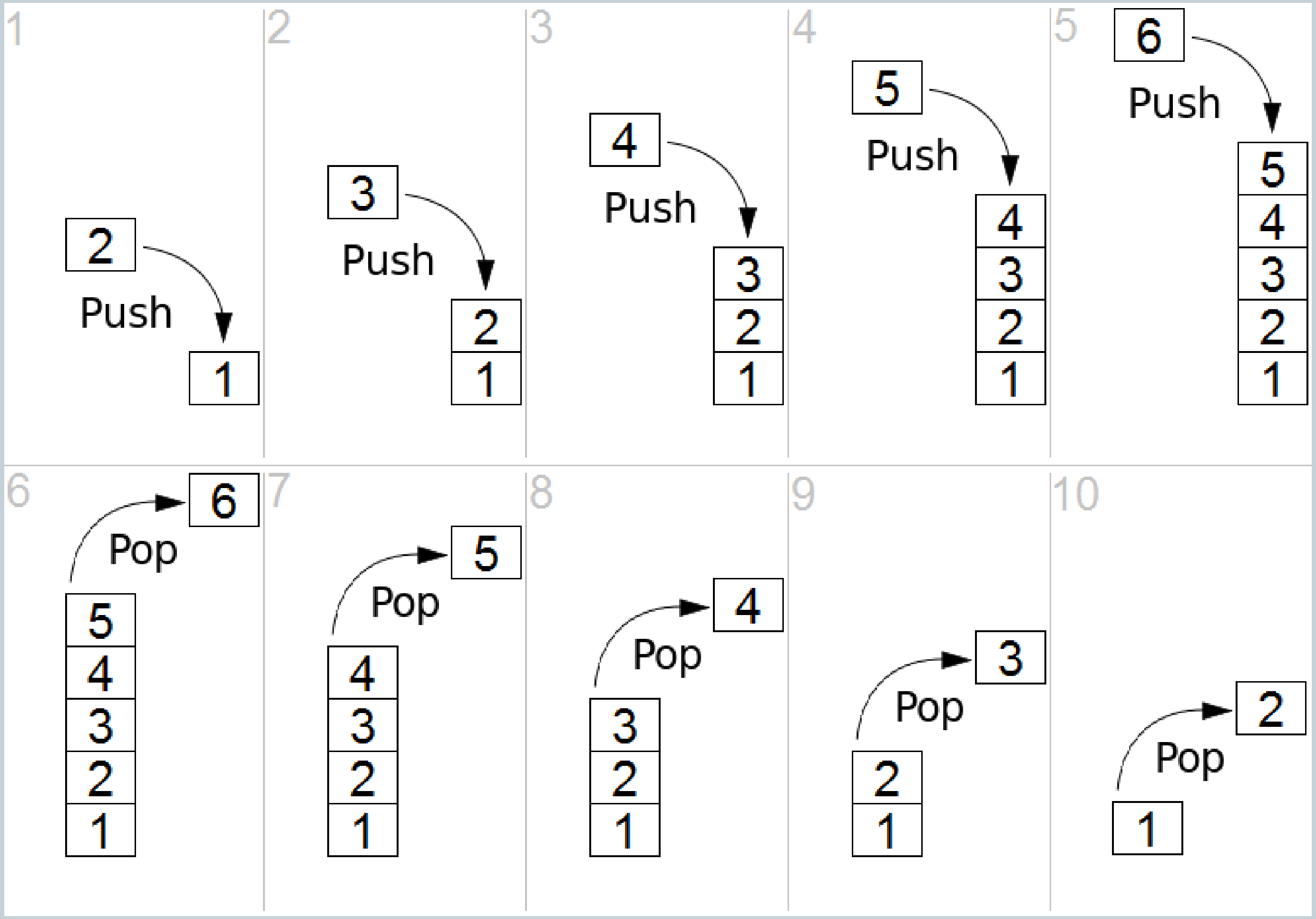
- A stack is said a **Last-In-First-Out (LIFO)** storage mechanism because the last item inserted is the first one to be removed.

Stacks: Operations

- **push** – to insert a data item on the top of the stack. If the stack is full and you cannot push new item. Theoretically, an ADT stack does NOT full, but the array implementing it does
- **pop** – to remove a data item from the top of the stack. If the stack is empty and you cannot pop an item more
- **peek** – to read value from the top of the stack without removing it (Somewhere called top)
- **size** – to return the number of items in the stack
- **isEmpty** – to check, is the stack empty? true – if the stack is empty
- **isFull** – to check, is the stack full? true – if the stack is full

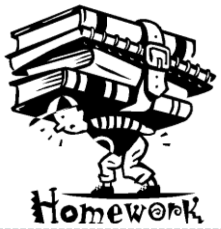
Try Stack Operations here: http://cathyatseneca.github.io/DSAnim/web/arraystack.html
https://www.youtube.com/watch?v=A_SobdSCY4Y

Stacks: Example of Push and Pop Operations



Stacks: Using Array for Implementing Stack (in C++)

```
#define DEF_SIZE 10                                          //Define Maximum Size of Array (stack)
class CMyStack{                                              //class CMyStack
  private:
      int StackMaxSize;                                      //Size of Stack Array
      int Array[ DEF_SIZE ];                                 //Stack Array
      int Top;                                               //Top of Stack
  public:
      void InitMyStack();                                    //Function initiation StackMaxSize and Top
      bool IsFull();                                         //Function to check, is the stack full? true - full
      void Push( int NewItem );                              //Function to push a new item to the top of stack
      bool IsEmpty()                                         //Function to check, is the stack empty? true - empty
      void Pop();                                            //Function to remove the top element of stack
      int Size();                                            //Function to return the number of elements in the stack
      int peek();                                       //Function to read value from the top of the stack without removing it
};
void CMyStack::InitMyStack(){
    StackMaxSize = DEF_SIZE;
    Top = -1;                                                //Initiation value of Top is -1 (means stack is empty)
};
bool CMyStack::IsFull(){
    if( Top == StackMaxSize-1 ) return true;                 //return true if the stack if true
    else return false;                                       //otherwise, return false
};
void CMyStack::Push( int NewItem ){
    if( !IsFull() ){                                         //check, if the stack is NOT full
        Top++;                                               //Move Top up (increase)
        Array[ Top ] = NewItem;                              //Add new item to the top of stack
        cout << "Push New Item Successfully";
    }
    else cout << "Stack is full";                            //if the stack is full
};
```



Create a Stack with the full operations: push, pop, peek, size, isEmpty, isFull

Read book of **Robert Lafore**, page: 125– 145 for next lecture

To be continued…