CHAPTER THREE

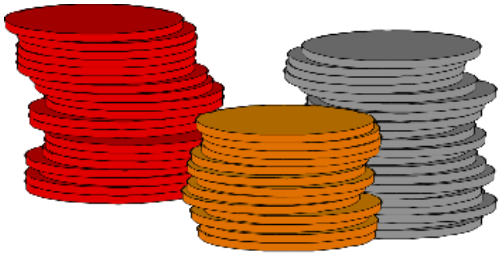
STACK

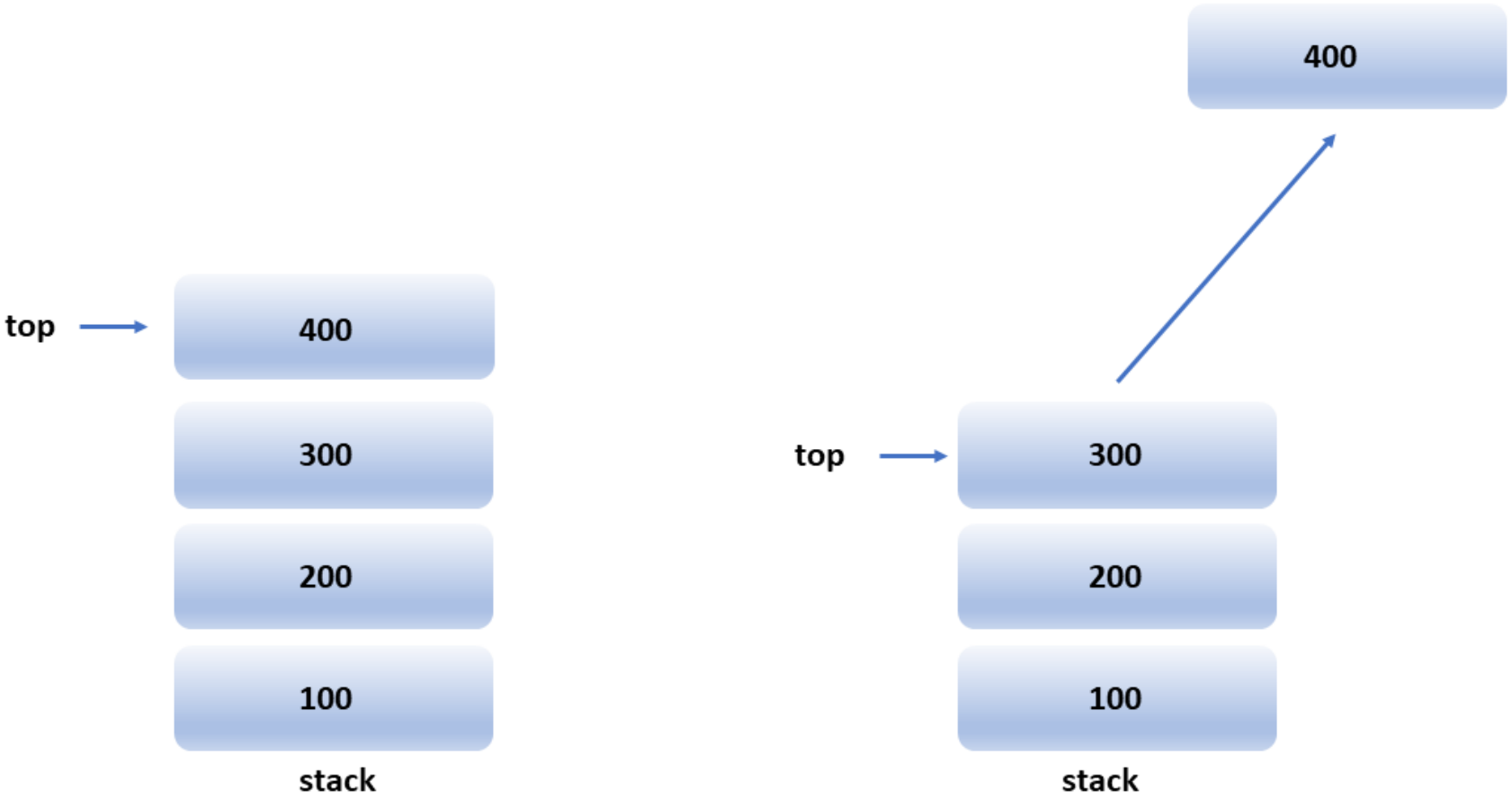
Introduction to Stack in Data Structures

- The stack data structure is a linear data structure that accompanies a principle known as LIFO (Last In First Out) or FILO (First In Last Out).
- Real-life examples of a stack are a deck of cards, piles of books, piles of money, and many more.

What is a stack?

- Stores a set of elements in a particular order
- Stack principle: LAST IN FIRST OUT/FILO
- = **LIFO**
- LILO: LAST ELEMENT TO BE ADDED IN WILL BE THE LAST OUT
- It means: the last element inserted is the first one to be removed.
- Example: Which is the first element to pick up?




top
400
300
200
100
stack
400
top
300
200
100
stack

PUSH() AND POP()

1. ***push (item):*** Inserts items on the top of the stack
2. ***pop ():*** Removes the top item from the stack

- Inserting an item is known as "**pushing**" onto the stack.
- "**Popping**" off the stack is synonymous with removing an item.

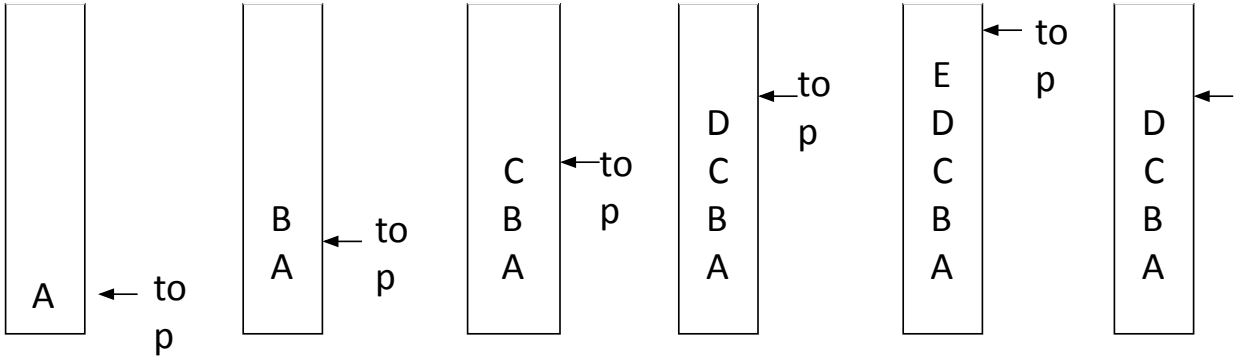
Push operation includes various steps, which are as follows :

1. Step 1: First, check whether or not the stack is full
2. Step 2: If the stack is complete, then exit
3. Step 3: If not, increment the top by one
4. Step 4: Insert a new element where the top is pointing
5. Step 5: Success

Pop Operation

- Step 1: First, check whether or not the stack is empty
- Step 2: If the stack is empty, then exit
- Step 3: If not, access the topmost data element
- Step 4: Decrement the top by one
- Step 5: Success

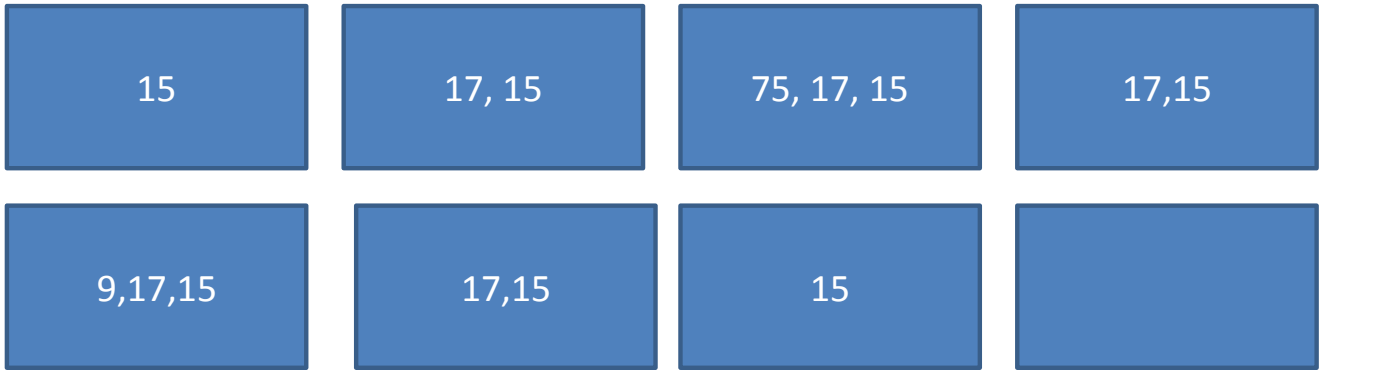
Last In First Out



Operations (methods) on stacks:

1. size (): Returns the number of items in the stack
2. empty (): Returns true if the stack is empty otherwise returns false
3. full(): Returns true if the stack is full
4. **ontop():** Returns the top element without removing it from the stack
5. **Peek()** operation returns the element currently at the top of the stack, but does not remove it.

Beginning with an empty stack , state its contents after performing the following operations in order. push(15), push(17), push(75), pop(),push(9), pop(),pop(),pop(),



Stack empty: *balanced*

EXAMPLE ONE: ADD THE FOLLOWING ON A STACK

push(15), push(17), push(75), pop(),push(9), pop(),pop(),pop(),

- ANSWER: EMPTY STACK

Stacks(3)

- A stack is an ADT that supports three main methods:
  - **push(S:***ADT,* **o:***element***):***ADT* - Inserts object *o* onto top of stack *S*
  - **pop(S:***ADT***):***ADT* - Removes the top object of stack *S*; if the stack is empty an error occurs
  - **top(S:***ADT***):***element* – Returns the top object of the stack, without removing it; if the stack is empty an error occurs

Stacks(4)

- The following support methods should also be defined:
  - **size(S:*ADT*):*integer*** - Returns the number of objects in stack *S*
  - **isEmpty(S:*ADT*): *boolean*** - Indicates if stack *S* is empty
- Axioms
  - **Pop(Push(S, v))** = **S**
  - **Top(Push(S, v))** = **v**

Operations on stacks

There are four basic operations, *stack*, *push*, *pop* and *empty*, that we define in this chapter.

The *stack* operation

The *stack* operation creates an empty stack. The following shows the format.

stack (stackName)

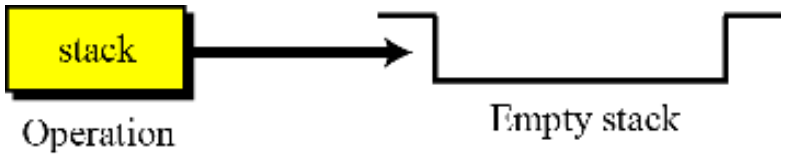


Figure 12.3 Stack operation

The *push* operation

The *push* operation inserts an item at the top of the stack. The following shows the format.

push (stackName, dataItem)

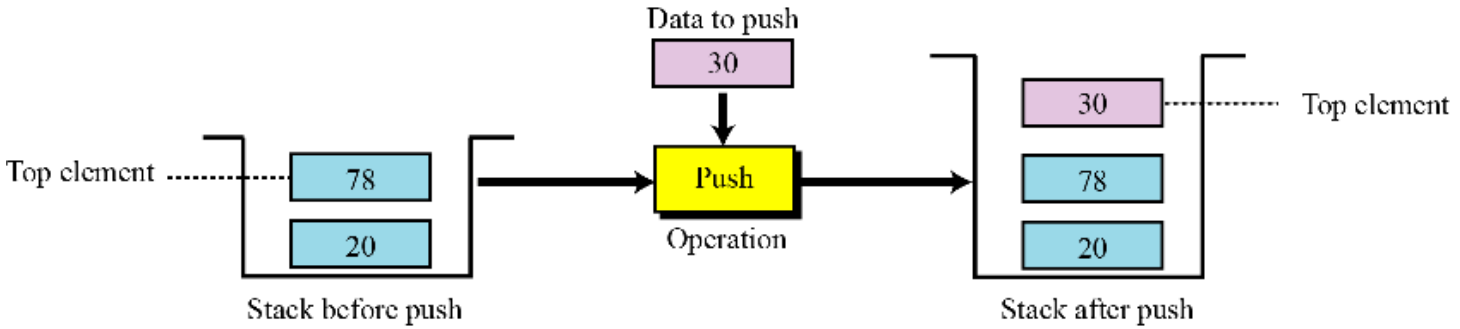


Figure 12.4 Push operation

The *pop* operation

The *pop* operation deletes the item at the top of the stack. The following shows the format.

pop (stackName, dataItem)

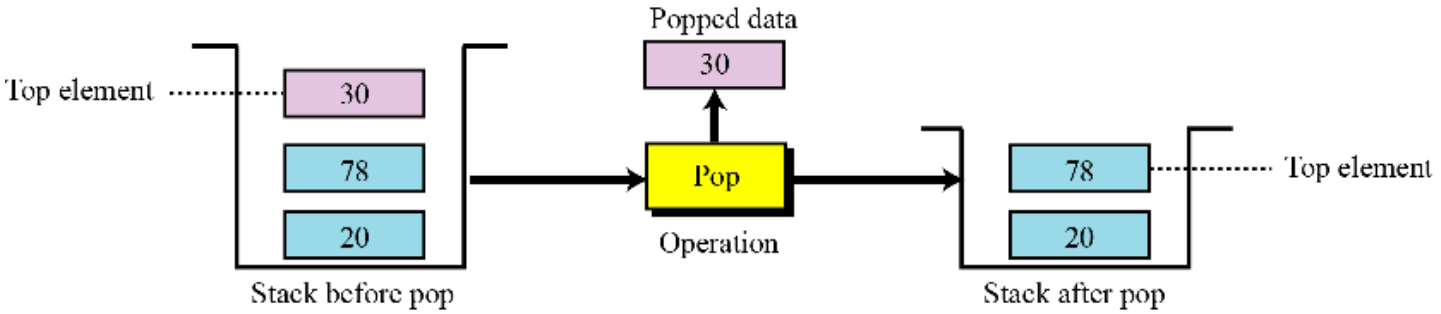


Figure 12.5 Pop operation

The *empty* operation

The *empty* operation checks the status of the stack. The following shows the format.

empty (stackName)

This operation returns true if the stack is empty and false if the stack is not empty.

An Array Implementation

- Create a stack using an array by specifying a maximum size *N* for our stack.
- The stack consists of an N-element array *S* and an integer variable *t*, the index of the top element in array S.

- Array indices start at 0, so we initialize *t* to -1

Example 12.1

Figure below shows a segment of an algorithm that applies the previously defined operations on a stack S.

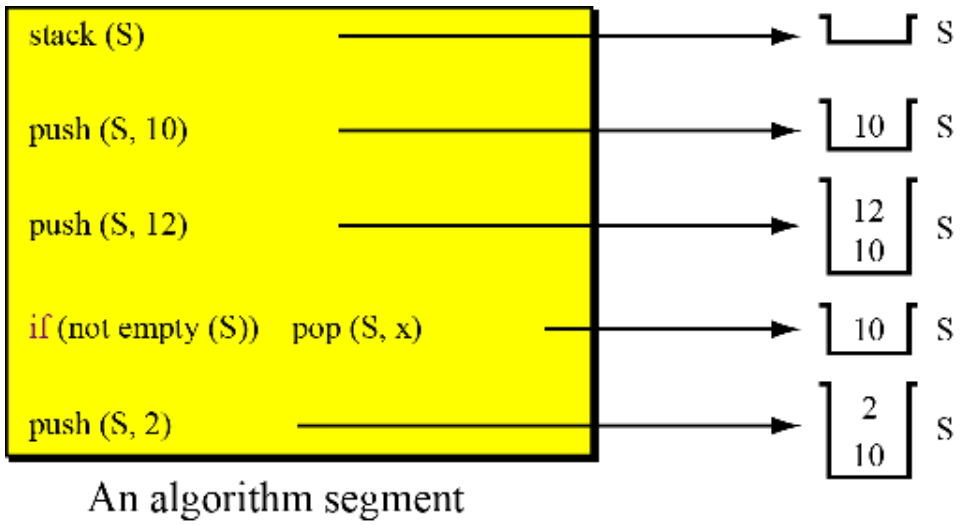


Figure 12.6 Example 12.1

TUTORIALS QUESTIONS

1. Define the term stack.
2. Explain how stack works
3. Stack are sometimes called LIFO. Explain
4. Explain the difference between pushing onto the stack and popping off the stack
5. State and explain the four basic operations of the stack and explain their corresponding syntax. *stack, push, pop* and *empty,*
6. Explain the applications of stack.

EXAMPLE USING IN C

```
1.  #include<stdio.h>
2.
3.  int main()
4.  {
5.      int a[100], i;
6.      printf("To pop enter -1\n");
7.      for(i = 0;;)
8.       {
9.          printf("Push ");
10.         scanf("%d", &a[i]);
11.           if(a[i] == -1)
12.           {
13.             if(i == 0)
```

```
1.  {
2.              printf("Underflow\n");
3.            }
4.          else
5.            {
6.              printf("pop = %d\n", a[--i]);
7.            }
8.          }
9.          else
10.         {
11.          i++;
12.         }
13.     }
14. }
```

Stack Applications

- **Real life**
  - Piling books on table
  - Piling Plates on a tray

In computing, stacks are used in various applications.

- One application is to support the *Backspace (on a PC) or delete (on a Mac) key on a computer keyboard.*
- *When you press this* key, the last letter typed (Last-In) is the one deleted first (First-Out).
- That is in the word, UNIVERSITY: Y WILL BE REMOVED FIRST

Stack applications

Stack applications can be classified into four broad categories: *reversing data*, *pairing data*, *postponing data usage* and *backtracking steps*. We discuss the first two in the sections that follow.

APPLICATION OF STACKS: cont.

- Reversing a String
- Palindrome Example
- BACKTRACKING
- MEMORY MANAGEMENT

Reversing a String

- Stacks can be used ***to reverse a sequence. For example, if a string "Computers" is entered by*** the user the stack can be used to create and display the reverse string "sretupmoC" as follows.
- KYAMBOGO: WRITING THE WORD IN REVERSE ORDER
- **WOULD BE: OGOBOMAYK**

PALINDROME EXAMPLE

- The strings where the **reading from the reverse and forward directions give the same word are called a palindrome**. For example, the string "radar" is an example for palindrome.
- Among many other techniques stack can be used to determine if a string is a palindrome or not.
- This is achieved by pushing all the letters of a given word into sack and checking if the letters popped are the same as the letter of the string.

Backtracking

- Backtracking is used in algorithms in which there are steps along some path (state) from some starting point to some goal.
- Play a game in which there are moves to be made (checkers, chess).

Backtracking

- In all of these cases, there are choices to be made among a number of options. We need some way to remember these decision points in case we want/need to come back and try the alternative
- Consider the maze. At a point where a choice is made, we may discover that the choice leads to a dead-end. We want to retrace back to that decision point and then try the other (next) alternative.

MEMORY MANAGENT

- DATA STORAGE IN RAM
- Memory management is an essential feature of the operating system, so the stack is heavily used to manage memory.

ASSIGNMENT ONE

- GROUP WORK: USING YOUR EXISTING GROUPS
- PREPARE 5 TO 10 SLIDES FOR EACH ALGORITHM
- DUE: 28/06/2022

Sorting Algorithms :

1. Selection Sort.
2. Bubble Sort.
3. Insertion Sort.
4. Merge Sort.
5. Quick Sort.