DATA STRUCTURES AND ALGORITHMS

Lecture 5: Stacks

Lecturer: Mohsin Abbas

National University of Modern Languages, Islamabad

STACKS

- A stack is a list in which insertion and deletion take place at the same end.
  - This end is called top.
  - The other end is called bottom.
- Stacks are known as LIFO (Last In, First Out) / FILO (First In, Last Out) lists.
  - The last element inserted will be the first to be retrieved and vice versa.
- Other names:
  - Push down list
  - Last In First Out (LIFO) list

STACKS

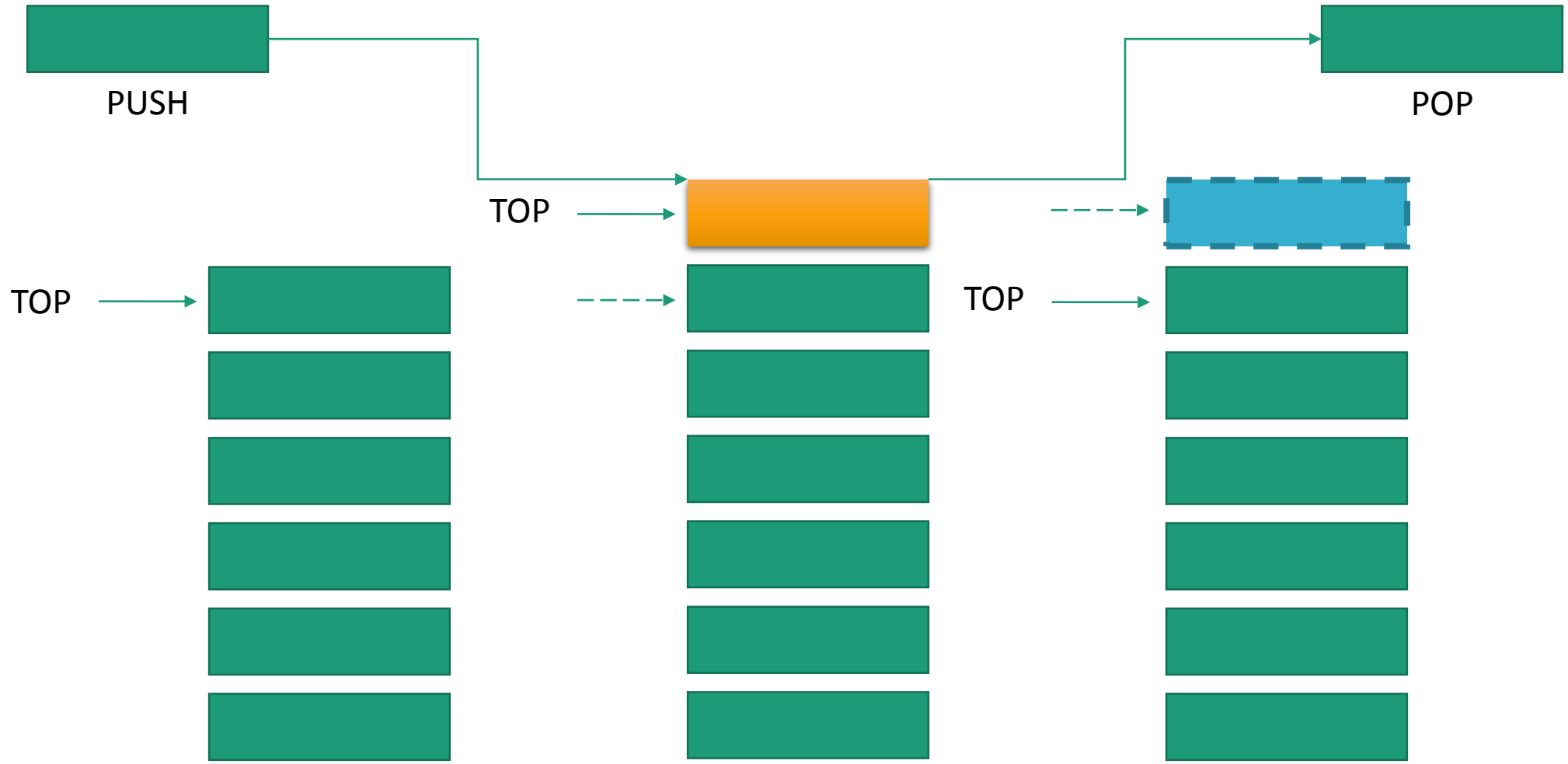
Examples:

- A stack of Plates
- Books on floor

STACKS

- Stack ADT emphasizes specific operations:
  - Uses a explicit linear ordering
  - Insertions and removal are performed individually
  - Inserted objects are pushed onto the stack
  - Top of the stack is most recently object pushed onto the stack
  - When an object is popped from the stack, the current top is erased

STACKS



STACK APPLICATIONS

- “Back” button of Web Browser
  - History of visited web pages is pushed onto the stack and popped when “back” button is clicked
- “Undo” functionality of a text editor
- Reversing the order of elements in an array
- Saving local variables when one function calls another, and this one calls another, and so on.

STACK OPERATIONS

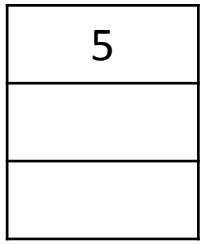
- Creating a stack
- Checking stack – either empty or full
- Insert (PUSH) an element in the stack
- Delete (POP) an element from the stack
- Access the top element
- Display the elements of stack

STACK OPERATIONS

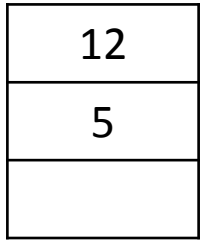
- MAKENULL(S)
  - Make Stack S be an empty stack
- TOP(S)
  - Return the top element of Stack
- POP(S)
  - Remove the top element of the stack
- PUSH(S, x)
  - Insert the element x at the top of stack
- EMPTY(S)
  - Return true if S is empty stack and return false otherwise

PUSH AND POP OPERATION OF STACK

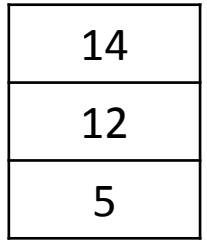
PUSH (5);



PUSH (12);

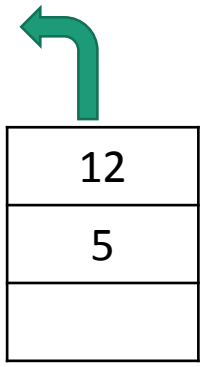


PUSH (14);



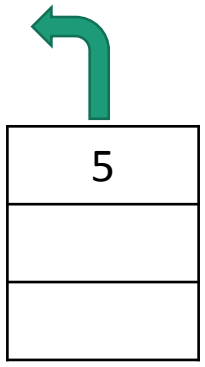
14

POP;



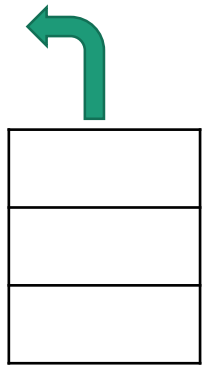
12

POP;



5

POP;



STACK RELATED TERMS

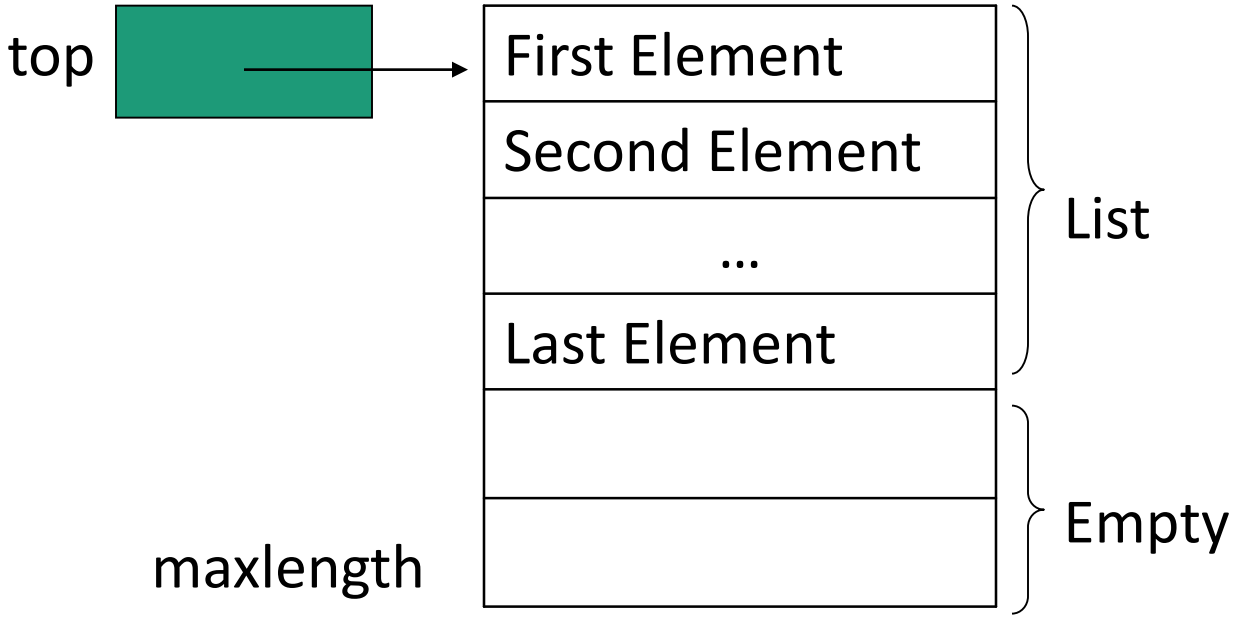
- Top
  - A pointer that points the top element in the stack.
- Stack Underflow
  - When there is no element in the stack, the status of stack is known as stack underflow.
- Stack Overflow
  - When the stack contains equal number of elements as per its capacity and no more elements can be added, the status of stack is known as stack overflow.

STACK IMPLEMENTATION

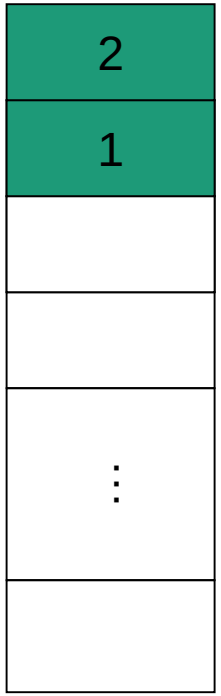
- Implementation of stack can be done in two ways
  - Static implementation
  - Dynamic Implementation
- Static Implementation
  - Stacks have fixed size, and are implemented as *arrays*
  - It is also inefficient for utilization of memory
- Dynamic Implementation
  - Stack grow in size as needed, and implemented as *linked lists*
  - Dynamic Implementation is done through pointers
  - The memory is efficiently utilize with Dynamic Implementations

STATIC IMPLEMENTATION

- Elements are stored in contiguous cells of an array.
- New elements can be inserted to the top of the list.

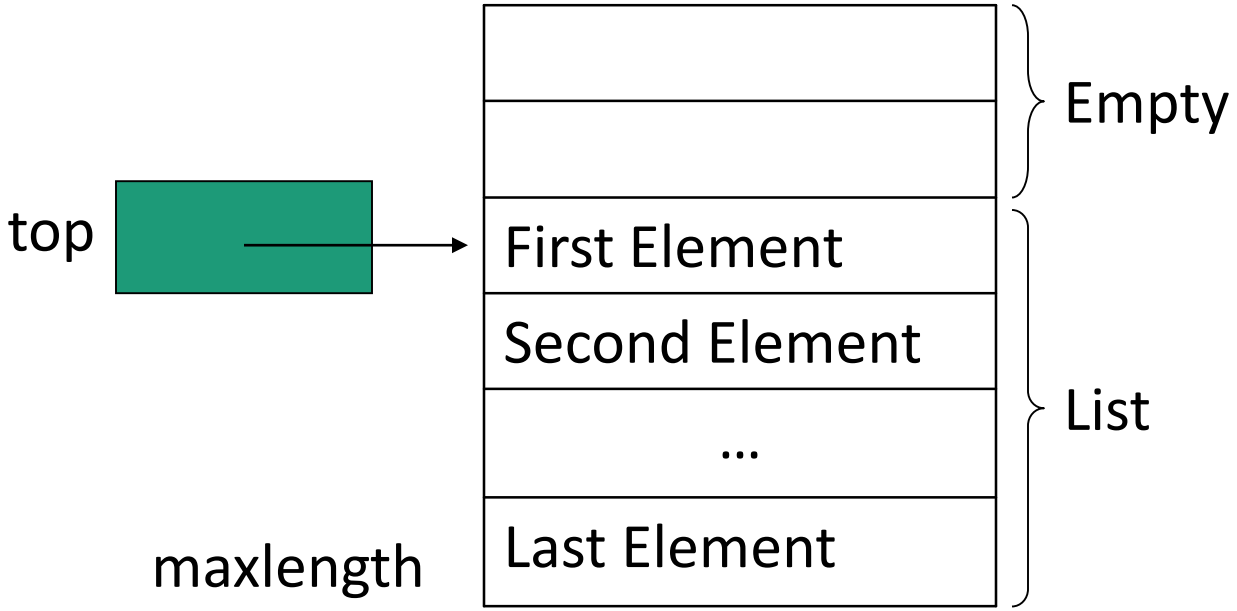


STATIC IMPLEMENTATION



- Problem with this implementation
  - Every PUSH and POP requires moving the entire array up and down.

STATIC IMPLEMENTATION



- Idea for better Implementation:
  - Anchor the top of the stack at the bottom of the array
  - Let the stack grow towards the top of the array
  - Top indicates the current position of the first stack element.

IMPLEMENTATION CODE

```
#include<iostream>
using namespace std;

#define max 10
int stack[max];
int top = -1;
void push();
void pop();
void display();
int main()
{
int option = 1;
char choice;
do
{
 cout<<"\nSelect stack
operation:\n\n1.PUSH\n2.POP\n3.EXIT\n\n";
 cout<<"Enter Choice: ";
 cin>>option;
```

```
switch(option)
 {
case 1:
  push();
  cout<<"\nElements in stack after PUSH
operation: \n";
  display();
  break;
  case 2:
  pop();
  cout<<"\nElements in stack after POP
operation: \n";
  display();
  break;
  case 3:
  exit(0);
  break;
  default:
  cout<<"Wrong Option. "; }
```

IMPLEMENTATION CODE

```
cout<<"\nDo you Repeat? Enter y/n or Y/N: ";
cin>>choice;
}while(choice=='y' || choice=='Y');
cout<<"\nProgram END"<<endl;
}
void push()
{
 int no;
 cout<<"\n\nHow many numbers u want to
insert:>> ";
 cin>>no;
 for (int j = 1; j<=no; j++)
 {
  int n;
  if (top>=max)
   cout<<"\nSTACK is FULL";
 else
  {
```

```
if (top<0)
   top = 0;
    cout<<"Element "<<j<<" in stack: ";
    cin>>n;
   stack[top++] = n;
  }
 }
}
void pop()
{
int e;
cout<<"\nHow many elements you want to POP:
";
cin>>e;
for (int k=1;k<=e;k++)
{
 if (top>=0)
 stack[top--];
} }
```

IMPLEMENTATION CODE

```
void display()
{
if (top<=0)
cout<<"\n\nSTACK is EMPTY";
else
for (int i = top-1;i>=0;i--)
cout<<stack[i]<<endl;
}

Output:



Select stack operation:

1.PUSH
2.POP
3.EXIT

Enter Choice: 1
```

```
How many numbers u want to insert:>> 7
Element 1 in stack: 5
Element 2 in stack: 14
Element 3 in stack: 12
Element 4 in stack: 72
Element 5 in stack: 9
Element 6 in stack: 110
Element 7 in stack: 1

Elements in stack after PUSH operation:
1
110
9
72
12
14
5
Do you Repeat? Enter y/n or Y/N: n
Program END
```

CONCLUSION

- In this lecture we have studied:
  - Stack Data Structure
  - Operations of Stack
  - Static Implementation of Stack

Question?