Chapter 19: Stacks and Queues

Kafi Rahman
Assistant Professor @ CS
Truman State University
// Distinct by Design

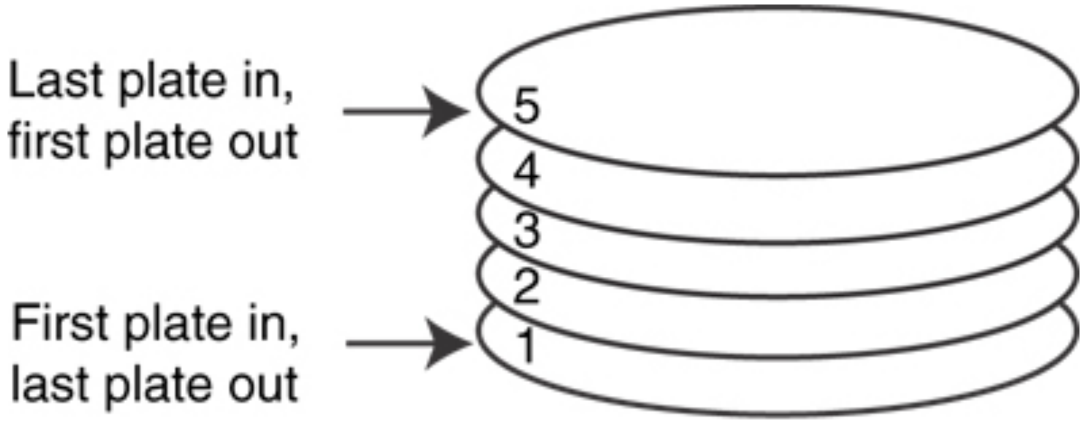
19.1

Introduction to the Stack ADT

Introduction to the Stack ADT

- Stack: a LIFO (last in, first out) data structure
- Examples:
  - plates in a cafeteria
  - return addresses for function calls
- Implementation:
  - static: fixed size, implemented as array
  - dynamic: variable size, implemented as linked list

A LIFO Structure

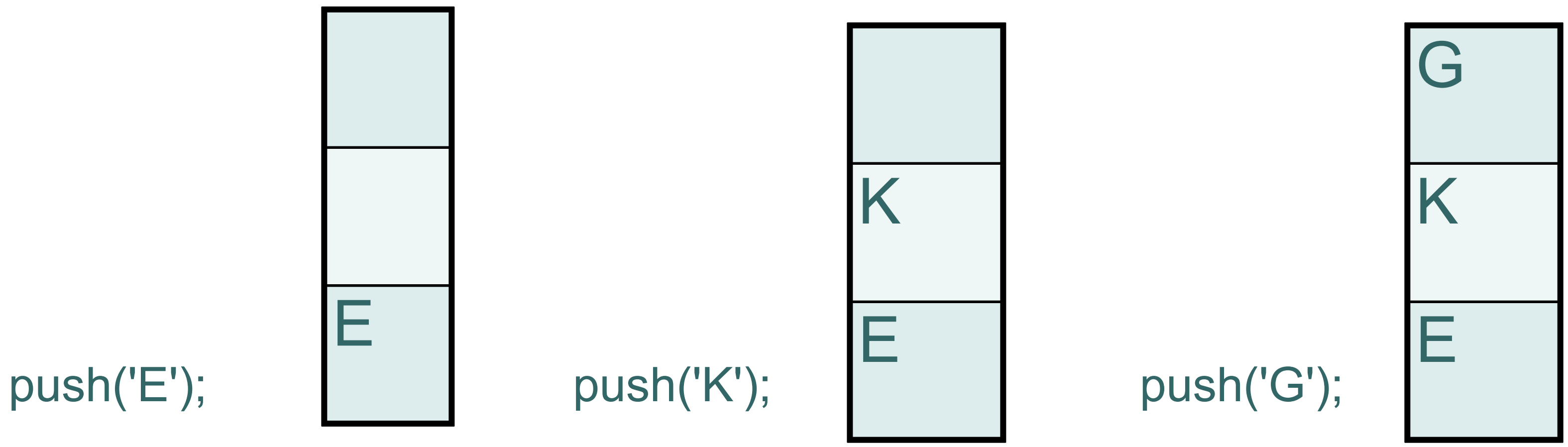


Stack Operations and Functions

- Operations:
  - push: add a value onto the top of the stack
  - pop: remove a value from the top of the stack
- Functions:
  - isFull: true if the stack is currently full, i.e., has no more space to hold additional elements
  - isEmpty: true if the stack currently contains no elements

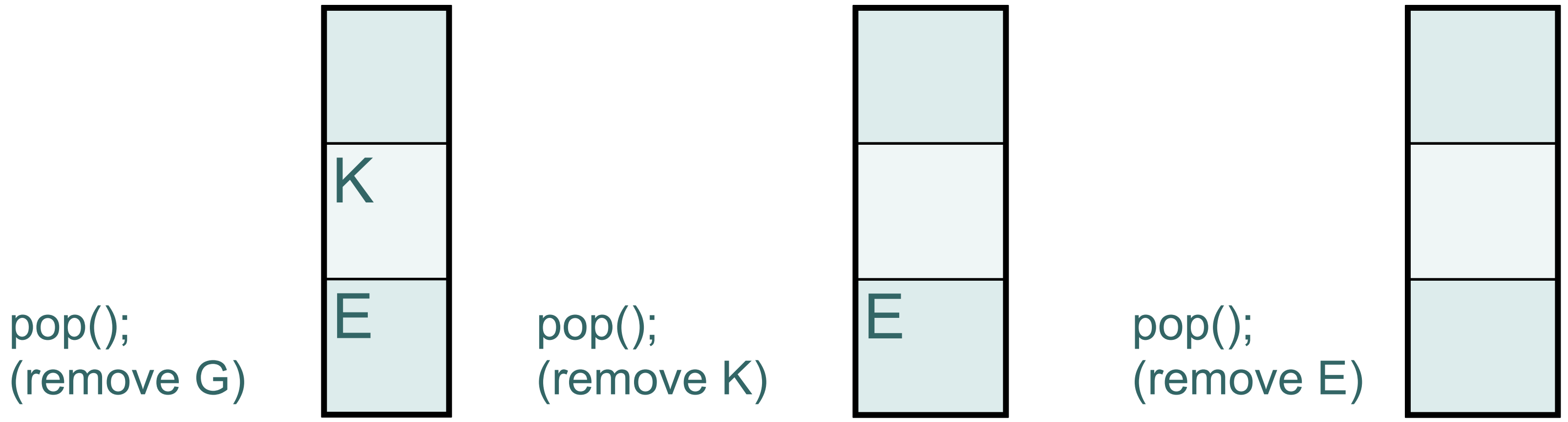
Stack Operations - Example

- A stack that can hold char values:



Stack Operations - Example

- A stack that can hold char values:



Contents of `IntStack.h`

```
// Specification file for the IntStack class
#ifndef INTSTACK_H
#define INTSTACK_H

class IntStack
{
private:
   int *stackArray;   // Pointer to the stack array
   int stackSize;     // The stack size
   int top;           // Indicates the top of the stack

public:
   // Constructor
   IntStack(int);

   // Copy constructor
   IntStack(const IntStack &);

   // Destructor
   ~IntStack();

   // Stack operations
   void push(int);
   void pop(int &);
   bool isFull() const;
   bool isEmpty() const;
};
#endif
```

19.2

Dynamic Stacks

Dynamic Stacks

- Grow and shrink as necessary
- Can't ever be full as long as memory is available
- Implemented as a linked list

Implementing a Stack

- Programmers can program their own routines to implement stack functions
- We are going to review the DynIntStack class implementation.
- Lastly, there are existing implementation of stack available in the STL. We are going to discuss a couple of examples of existing STL classes

Dynamic Stack: adding an element

19.3

The STL stack Container

The STL stack container

- Stack template can be implemented as a vector, a linked list
- Implements push, pop, and empty member functions
- Implements other member functions:
  - size: number of elements on the stack
  - top: reference to element on top of the stack

Defining a stack

- Defining a stack of chars, named cstack, implemented using a vector:
  - stack< char, vector<char>> cstack;
- implemented using a list:
  - stack< char, list<char>> cstack;

- When using a compiler that is older than C++ 11, be sure to put spaces between the angled brackets that appear next to each other.

```
stack< char, vector<char> > cstack;
```