CS 214 – DATA STRUCTURES FALL 2015

Lecture 2 – Stacks

Group **B**

25th September 2017

Dr. Mai Hamdalla

mai@fci.helwan.edu.eg

House Keeping

- Lecture and Lab Attendance
- Syllabus
- Piazza
  - Signed up?
  - Questions/Concerns
- Implementation Environment
  - Code Blocks

Abstract Data Type (ADT)

- Is a type defined in terms of its **data items** and **associated operations**, not it's implementation.

ADT = Organized data + Operations

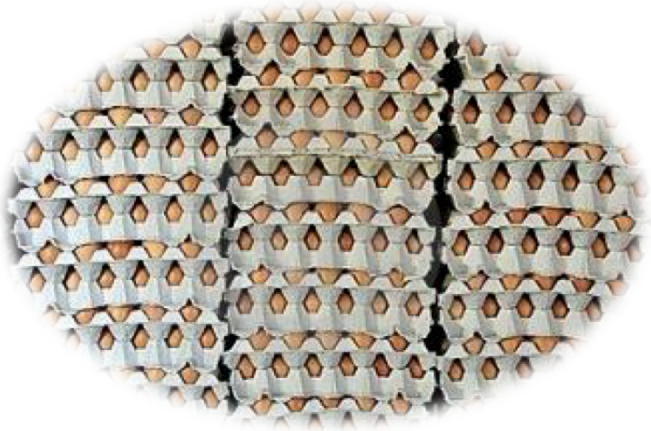
Why ADT?

- The implementation of the component can be changed without affecting some other component.

- The user *only* has to study the **interface** at a much higher level, rather than having to understand the detailed implementation of the set operations.

STACKS

What is a Stack?



What is a Stack?

- Stack is LIFO Structure
- Stack is Ordered List of Elements of **Same Type**.
- Stack is a Linear List.
- In Stack, all Operations are permitted at only one end called **Top.**

Stack Example -

- Trace the following example showing the how the **stack** would look and the value of **top** after each operation:
  - Add (5)
  - Add (-17)
  - Delete
  - Add (8)
  - Delete
  - Add (-2)

Operations Performed on Stack

- Create the stack, leaving it empty.
- Determine whether the stack is empty or not.
- Determine whether the stack is full or not.
- Push a new entry onto the top of the stack
- Pop the entry off the top of the stack.

STACK CONTIGUOUS IMPLEMENTATION

Stack Struct

```
#define MAX 10

typedef char entry_type;
typedef struct{
    int top;
    entry_type entry[MAX];
} StackType;
```

Stack Initialization

- **Pre:** None.
- **Post:** The stack is initialized to be empty.

```
void create_stack(StackType *s){
    s->top = -1;
}
```

Stack Empty

- **Pre:** The stack is initialized.
- **Post:** If the stack is empty, (1) is returned, otherwise (0) is returned.

```
int is_stack_empty(StackType s){
    return (s.top == -1);
}
```

Stack Full

- **Pre:** The stack is initialized.
- **Post:** If the stack is full, (1) is returned.
  Otherwise (0) is returned.

```
int is_stack_full (StackType s){
    return (s.top == MAX-1);
}
```

Stack Push Operation

- **Pre:** The stack is initialized and is not full.
- **Post:** Item is added to the top of the stack.

```
void push(EntryType item,
                    StackType *s){
    s->entry[++s->top] = item;
}
```

```
s->top++;
s->entry[s->top]=item;
```

Stack Push Operation

- **Pre:** The stack is initialized.
- **Post:** If the stack is not full, item is added to the top of the stack. Otherwise, an error message is displayed and the stack is left unchanged.

```
void push(EntryType item,
                    StackType *s){
    if (s->top == MAX-1)
         printf("Error: Stack overflow")
    else
         s->entry[++s->top] = item;
}
```

Stack Pop

- **Pre:** The stack is initialized and is not empty.
- **Post:** The top element of the stack is removed from it and is assigned to item.

```
void pop(EntryType *item,
                 StackType *s){
    *item = s->entry[s->top--];
}
```

Example

- **Write a C program to read a line of text from the user and print it back on the screen in reverse order.**

Example Soln.

```
StackType stack;
//Initialize the stack
create_stack(&stack);
//read character from user
item = getchar();
while (!is_stack_full(stack)
                        && item != '\n'){
   //Push each item onto the stack
   push(item, &stack);
   //read character from user
    item = getchar();
}
```

Example Soln. Cont'd

```
while (!is_stack_empty(stack)){
    //Pop an item from the stack
    pop(&item, &stack);
    //Print character on screen
    putchar(item);
  }
```

YOUR RESPONSIBILITY

Your are responsible for:

- Reading and Studying chapter covering Stacks (Chapter 3 in Data Structures and Algorithm Analysis in C)
- Searching for online material/problems regarding Stacks and ADTs.
- Getting familiar with: https://www.gnu.org/prep/standards/html_node/Writing-C.html

CS 214 – DATA STRUCTURES FALL 2015

Lecture 2 – Stacks

Group **B**

25th September 2017

Dr. Mai Hamdalla

mai@fci.helwan.edu.eg