Data Structures & Algorithms

Stack

Dr. Iftikhar Ahmad
ia@uetpeshawar.edu.pk

Department of CS & IT
University of Engineering & Technology, Peshawar

Overview

1. Stack
   - Motivation
2. Introduction
   - Introduction
   - Operations on Stack
   - Representation of Stack
3. Push Operation
   - Definition
   - Push :: C++ Code
4. Pop Operation
   - Definition
   - Pop :: C++ Code
5. Stack :: Complete C++ Code
6. Application of Stack
   - Evaluating Mathematical Equations
   - Evaluating Postfix Notation
   - Converting Infix to Postfix notation
7. Acknowledgement & Questions
   - Acknowledgement
   - Questions

Overview

1. Stack
   - Motivation
2. Introduction
   - Introduction
   - Operations on Stack
   - Representation of Stack
3. Push Operation
   - Definition
   - Push :: C++ Code
4. Pop Operation
   - Definition
   - Pop :: C++ Code
5. Stack :: Complete C++ Code
6. Application of Stack
   - Evaluating Mathematical Equations
   - Evaluating Postfix Notation
   - Converting Infix to Postfix notation
7. Acknowledgement & Questions
   - Acknowledgement

Motivation

- Consider a 4-player game of cards, in which each player is has 10 cards to start with.
- Upon her turn a player retrieves the top most card from the deck (the remaining cards) and can replace it with one of her cards. The extra card is placed (upside down) in a separate deck.
- The winning player must have the following combination;
  - Have two combinations of cards from the following two;
    1. Three cards of consecutive *ranks* of same *suit*
    2. Three cards of the same *ranks* of different *suits*
  - Have one combination of cards from the following two;
    1. Four cards of consecutive *ranks* of same *suit*
    2. Four cards of the same *ranks* of different *suits*
- Which *data structure* will be ideal to represent the *deck* allowing only RETRIEVAL and REMOVING of the top most element. INSERTION should also be allowed at top.
- Some possibilities;
  - Arrays.
  - Linked List

Motivation

- Problems with using ARRAYS for *deck*.
- Problems with using LINKED LIST for *deck*.

Solution

- Based on the requirement, we need a data structure that;
  - Allows INSERTION at top location.
  - Allows DELETION at top location.
  - Does not allow access to other locations other than *top*.
  - i.e., the data structure works based on the principle of LIFO - Last In First Out.
- STACK is the data structure that fulfils all the requirements.

Overview

1. Stack
    - Motivation
2. Introduction
    - Introduction
    - Operations on Stack
    - Representation of Stack
3. Push Operation
    - Definition
    - Push :: C++ Code
4. Pop Operation
    - Definition
    - Pop :: C++ Code
5. Stack :: Complete C++ Code
6. Application of Stack
    - Evaluating Mathematical Equations
    - Evaluating Postfix Notation
    - Converting Infix to Postfix notation
7. Acknowledgement & Questions
    - Acknowledgement

Introduction

Stack :: Definition

Stack is a data structure that works on the principle of LIFO - Last In First Out.

Informal Examples

- Stack of plates.
- Stack of books.
- Deck of cards.

Operations on Stack

Permissible Operations on Stack

- PUSH - Inserting a new element.
- POP - Removing the top most element.

Representation of Stack

```
struct myStack{
  private:
    int stack[N];
    int top = -1;
};
```

Overview

1. Stack
   - Motivation
2. Introduction
   - Introduction
   - Operations on Stack
   - Representation of Stack
3. Push Operation
   - Definition
   - Push :: C++ Code
4. Pop Operation
   - Definition
   - Pop :: C++ Code
5. Stack :: Complete C++ Code
6. Application of Stack
   - Evaluating Mathematical Equations
   - Evaluating Postfix Notation
   - Converting Infix to Postfix notation
7. Acknowledgement & Questions
   - Acknowledgement

Push :: Definition

- PUSH operation is defined as inserting element at the top of a stack.
- A variable TOP is used to keep track of elements added so far.

Push Operation :: C++ Code

```
/*
 * Insert the value at the top
 * and increments the top
*/
bool push(int value){
    if(isFull())
      return false;

     //First increments the value of
        top
     //then store value at the top
        position
     stack[++top] = value;
     return true;
 }
```

Overview

1. Stack
   - Motivation
2. Introduction
   - Introduction
   - Operations on Stack
   - Representation of Stack
3. Push Operation
   - Definition
   - Push :: C++ Code
4. Pop Operation
   - Definition
   - Pop :: C++ Code
5. Stack :: Complete C++ Code
6. Application of Stack
   - Evaluating Mathematical Equations
   - Evaluating Postfix Notation
   - Converting Infix to Postfix notation
7. Acknowledgement & Questions
   - Acknowledgement

Pop :: Definition

- POP operation refers to removing the top most element from the stack.
- The variable TOP is decremented and the top most element is returned.

Selection Sort :: C++ Code

```
/*
 * Removes the top most element
 * and decrements the top
 */
int pop(){
   if(isEmpty())
     return -1;

   //returns the top most element
   //and decrements the value of top
   return stack[top--];
}
```

Overview

1. Stack
   - Motivation
2. Introduction
   - Introduction
   - Operations on Stack
   - Representation of Stack
3. Push Operation
   - Definition
   - Push :: C++ Code
4. Pop Operation
   - Definition
   - Pop :: C++ Code
5. Stack :: Complete C++ Code
6. Application of Stack
   - Evaluating Mathematical Equations
   - Evaluating Postfix Notation
   - Converting Infix to Postfix notation
7. Acknowledgement & Questions
   - Acknowledgement

Stack :: C++ Code

```
const int N = 5;
struct myStack{
private:
        int stack[N];
        int top = -1;

public:

         // Checks if the stack is
            empty
        bool isEmpty(){
                return (top==-1);
        }

         // Checks if the stack is full

        bool isFull(){
                return(top==N-1);
        }
```

Stack :: C++ Code (cont...)

```
         // Insert the value at the top
         // and increments the top
        bool push(int value){
                if(isFull())
                        return false;

                //First increments the
                   value of top
                //then store value at
                   the top position
                stack[++top] = value;
                return true;
        }
         // Removes the top most
            element
         // and deceremetns the top
        int pop(){
                if(isEmpty())
                        return -1;

                //returns the top most
                   element
                //and decrements the
                   value of top
                return stack[top--];
        }
};
```

Overview

1. Stack
   - Motivation
2. Introduction
   - Introduction
   - Operations on Stack
   - Representation of Stack
3. Push Operation
   - Definition
   - Push :: C++ Code
4. Pop Operation
   - Definition
   - Pop :: C++ Code
5. Stack :: Complete C++ Code
6. Application of Stack
   - Evaluating Mathematical Equations
   - Evaluating Postfix Notation
   - Converting Infix to Postfix notation
7. Acknowledgement & Questions
   - Acknowledgement

Application of Stack

- Function Calls.
- Evaluating Mathematical Equations
    - Converting from prefix to postfix notation.
    - Evaluating postfix expressions.
- Conversion from decimal to binary.

Evaluating Mathematical Equations

- Mathematical equations can be represented in different forms;
  - Infix Notation
    - Operators is written in between operands.
    - e.g., $A + B$
    - For enforcing precedence of an operation, () are used.
  - Postfix Notation
    - Operands precedes operators.
    - e.g., $AB+$
    - Equation is evaluated from left to right, without the need of precedence operators such as ().
    - Also called as RPN - Reverse Polish Notation.
  - Prefix Notation
    - Operators precedes operands.
    - e.g., $+AB$
    - Also called as PN - Polish Notation.

Basic Idea

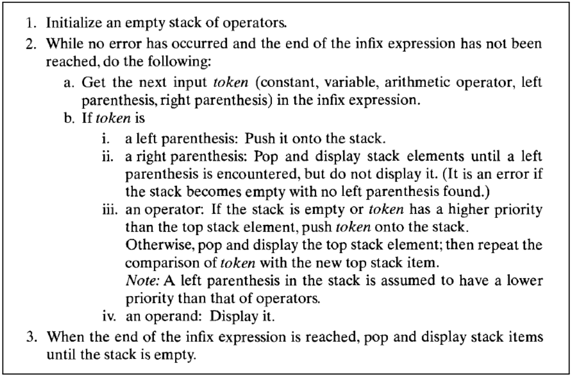
Evaluating Postfix Notation

1. Scan the expression from left to right
    1. If an operand is encountered : Push it onto stack.
    2. If an operator is encountered : Pop the required number of operands, perform the operation and push the result back onto stack.
2. After the expression is scanned, the result is stored at top of the stack.

Example

Expression	Stack	Comments
2 4 * 9 5 + -	2	Push 2 onto the stack
4 * 9 5 + -	4 2	Push 4 onto the stack
* 9 5 + -	8	Pop 4 and 2 from the stack, multiply, and push the result back onto the stack.
9 5 + -	9 8	Push 9 onto the stack
5 + -	5 9 8	Push 5 onto the stack
+ -	14 8	Pop 5 and 9 from the stack, add, and push the result back onto the stack.
-	-6	Pop 14 and 8 from the stack, subtract, and push the result back onto the stack.
(End of String)	-6	Resultant value is at top

Infix to Postfix Conversion



1. Initialize an empty stack of operators.
2. While no error has occurred and the end of the infix expression has not been reached, do the following:
    a. Get the next input *token* (constant, variable, arithmetic operator, left parenthesis, right parenthesis) in the infix expression.
    b. If *token* is
        i. a left parenthesis: Push it onto the stack.
        ii. a right parenthesis: Pop and display stack elements until a left parenthesis is encountered, but do not display it. (It is an error if the stack becomes empty with no left parenthesis found.)
        iii. an operator: If the stack is empty or *token* has a higher priority than the top stack element, push *token* onto the stack.
        Otherwise, pop and display the top stack element; then repeat the comparison of *token* with the new top stack item.
        Note: A left parenthesis in the stack is assumed to have a lower priority than that of operators.
        iv. an operand: Display it.
3. When the end of the infix expression is reached, pop and display stack items until the stack is empty.

Figure: Prefix to Postfix Algorithm [1]

[1] Nyhoff, Larry. *ADTs, Data Structures, and Problem Solving with C++*. [Chapter 7]

Example

Expression	Stack	Output	Comments
7 * 8 - (2+3)		7	Display 7
* 8 - (2+3)	*	7	Push *
8 - (2+3)	*	7 8	Display 8
- (2+3)		7 8 *	Pop & Display *
	-	7 8 *	Push -
(2+3)	(-	7 8 *	Push (

Expression	Stack	Output	Comments
2+3)	(-	7 8 * 2	Display 2
+3)	+ (-	7 8 * 2	Push +
3)	+ (-	7 8 * 2 3	Display 3
)	(-	7 8 * 2 3 +	Pop & Display +
	-	7 8 * 2 3 +	Pop (
		7 8 * 2 3 + -	Pop & Display -

Overview

1. Stack
   - Motivation
2. Introduction
   - Introduction
   - Operations on Stack
   - Representation of Stack
3. Push Operation
   - Definition
   - Push :: C++ Code
4. Pop Operation
   - Definition
   - Pop :: C++ Code
5. Stack :: Complete C++ Code
6. Application of Stack
   - Evaluating Mathematical Equations
   - Evaluating Postfix Notation
   - Converting Infix to Postfix notation
7. Acknowledgement & Questions
   - Acknowledgement

Acknowledgement

Source Acknowledgement

The material (including figures and examples) presented in this lecture is adopted from the following book;

- Nyhoff, Larry. *ADTs, Data Structures, and Problem Solving with C++*. [Chapter 7]

Questions

Questions?