Kyle Knudson

CSCI 591

Cs301188

Section 1

03/21/18

Project 8

Project 8 - Design Document

Introduction:

When evaluating algebraic expressions a traditional order of operations is used to correctly evaluate the expression. However, this only works when the information is presented in a very specific order and the digits and symbols are in the correct positions. This method of presenting arithmetic expressions is referred to as infix notation. In computing though, there are various different notations that an arithmetic expression can be presented in. In this project expressions in a file are listed in prefix notation. Prefix notation is where the operators are presented in a manner in which they precede the operands. This notation is best evaluating using a stacked based algorithm that contains one stack that holds both operands and operators. For this program, the user is asked to input a data file that consists of the expressions that the user wishes to be evaluated. The file contains one expression per line and the end of the line signals the end of the expression. After the file is opened and read, the algorithm evaluates the expressions and displays the values to the terminal.

Data Structures:

The data structures used in this class is a stack that is pointer based rather than an array-based stack like in project 7. This program also utilizes structs that hold either an integer value or a character value depending on what is read from the file.

Functions:

Read(); - This function reads the data from the file one character at a time so that it is able to be stored in the stack object.

Apply(); - This function takes in an operator and two operands as parameters and applies the operator to the operands and determines the value of the small operation.

Display(); - This function displays the value of the expression to the terminal.

Pop(); - this function reads the value that is at the top of the stack and decrements the used data member that keeps track of how many values are on the stack.

chPop(); - this function reads the value that is at the top of the stack and decrements the used data member that keeps track of how many values are on the stack.

Push(); - This function stores the value onto the top of the stack object and increments the used data member that keeps track of how many values are on the stack.

Peek(); - This function looks to see what the next value on the stack is going to be.

Stack(); - This function is the constructor for the stack object.

~Stack(); - This function is the destructor for the stack object and gives back dynamic data.

Is\_top\_data() – shows what the type of the top value is.

displayStack() – prints out the contents of the stack in order

The Main program:

The main function for this program is responsible for prompting the user for the file that contains the expressions that the user wishes to be evaluated. Once the file has been opened, the expressions are read one character at a time and pushed onto the stack object so that the expression is able to be evaluated. After, all of the expressions have been read, the algorithm pops the values of the stack and the operators are applied to operands. This is done by a function in the main program that applys the operator to the two operands that are passed to the function as parameters. After this is done the value that remains is the final value of the expression. This value is then displayed to the user on the terminal, after all of the stack is processed the program terminates.