DATA STRUCTURE

PARENTHESIS MATCHING

INFIX TO POSTFIX

ARİFE GÜL YALÇIN

B1605.090054

CONTENTS

..... 1

1. INTRODUCTION..... 3

1.1. LIBRARIES..... 3
1.2. PUSH() 3
1.3. POP()..... 3
1.4. PRECEDENCE() 4
1.5. MAIN() 4

SUMMARY

I have written an application for Infix to Postfix expression using C language. Here I have shown how to implement infix to postfix for an expression written with parenthes types such as (), [] and {}. When the parentheses are written incorrectly, the program fails and does not perform the operation.

1. INTRODUCTION

First of all, I created a Stack data structure. Inside this structure is an array and a top (int top) value that specifies the MAXSIZE (int stk [MAXSIZE]) of the stack as the int value.

The default initial value for top is -1.

Then I defined void push () and int pop () functions.

1.1. Libraries

- **stdio.h**
- **string.h**
- **ctype.h →** Used for isalnum () function
- **MAXSIZE 50 →** Maximum array size for stack

1.2. Push()

The current character is a starting bracket (‘(‘ or ‘{‘ or ‘[‘) .

Function called to add value to expression..

- ✓ If the stack is full, I can't add elements.
- ✓ So I first checked if the directory is full.
- ✓ If the array is not full, a new element can be added.
- ✓ For this, I assigned the number to the index by increasing the ball value from -1 at the beginning.

1.3. Pop()

The current character is a closing bracket (‘)’ or ‘}’ or ‘]’).

Function called to extract value from expression.

- ✓ First I checked if the stack is empty.
- ✓ If the array is not empty, the element can be omitted.
- ✓ This can only be done by decreasing the top value by 1.

1.4. **Precedence()**

I wrote such a function for priority order.

‘*’ and ‘/’ values are before ‘+’ and ‘-‘ values are later expressions.

1.5. **Main()**

Firstly, I defined expression[100] and I also defined a pointer named 'exp' and I did it to get it from the expression value that the user has entered.

IF

I checked whether the 'e' parameter value passed in the Expression is a letter or a number in the alphabet. And the program printed the expression.

ELSE IF

- ➔ Expression is pushed if it starts with parentheses
- ➔ If Expression is the closing parenthesis, pops the values up to the opening brackets

ELSE

Expression values are subtracted from the array until it reaches top.Pop function is applied.

push(*exp); Then the entire expression is pushed.

I wrote a while loop for the expression to subtract values until the top value reaches -1, i.e. until the array ends.

```
Enter the Expression: {[9+2*(6/3)-1]}
9 2 6 3 / * + 1 -
--------------------------------
Process exited after 30.68 seconds with return value 0
Press any key to continue . . .
```

True Expression

```
Enter the Expression: {[5-2)}
```

```
C:\Users\HP\Desktop\para.exe
```

False Expression