Apex College

BCIS Program

Affiliated to Pokhara University

Data Structure & Algorithms
Lab Report
5. Conversion of
Infix to postfix &
Infix to prefix

Date: 30-05-2022

Submitted by:
Ishwor Shrestha
Roll no.: 2018-BCIS-414

Submitted to:
Pravakar Ghimire, &
Anmol Shrestha
Apex College

#Lab 5 Objectives

- To convert infix expression into postfix expression.
- To convert infix expression into prefix expression
- To implement concept of stack to convert infix expression into postfix and prefix expressions.

#Introduction

Infix expressions are the normal algebric ~~op~~ expression which is easily ~~reable~~ readable for human being. Whereas, prefix and postfix expressions are re-arrangement of infix expressions which is easily readable operations as compare to infix. In post and pre fix expressions there is less or no confusion in operand.

We use stack to change infix operation into postfix or ~~prefix~~ prefix operations. We use precedence order of operators to maintain execution priority with following ~~ord~~ order:

1. $, ^
2. /, *
3. ~~+~~ +, -

- Infix : A+B
- Prefix : +AB
- Postfix : AB+

#Programs to convert infix into prefix and postfix:

* For ~~postfix~~ postfix expression

```
#include<stdio.h>
#include<stdlib.h>

char stack[100];
int top =-1;
void push(char x){
    stack[++top] = x;
}
```

```
char pop (){
    if (top = = -1)
        return -1;
    else
        return stack [top--];
}

int priority (char x){
    if (x = = '(')
        return 0;
    if (x== '+' || x = = '-')
        return 1;
    if (x= '*' || x = '/')
        return 2;
    return 0;
}

int main(){
    char exp [100];
    char *e, x;
    printf ("Enter the expression: ");
    scanf ("%s", exp);
    printf ("\n");

    e = exp;

    while (*e! = '\0') {
        if (isalnum (*e))
            printf ("%c", *e);
        else if (*e = = '(')
            push (*e);
        else if (*e = = ')'){
```

CamScanner

```
            while ((x = pop ()) != '(')
                printf ("%c", x);
        }
        else {
            while (priority (stack [top]) >= priority (*e))
                printf ("%c", pop ());
            push (*e);
        }
        e++;
    }
    while (top != -1) {
        printf("%c", pop());
    } return 0;
}
```