ГУО «БГУИР»

Факультет информационных технологий и управления

Кафедра систем управления

Отчет по

Лабораторной работе №4

Обратная польская запись

Подготовил:

Студент гр.222401

Бык А.А.

Проверила:

Семижон Е.А.

Минск 2023

Цель: изучить правила формирования постфиксной записи арифметических выражений с использованием стека.

Вариант №3.

Написать программу формирования ОПЗ и расчета полученного выражения. Разработать удобный интерфейс ввода исходных данных и вывода результатов. Работу программы проверить на конкретном примере (табл. 5.1).

Код:

#include<stdio.h>

#include<stdlib.h>

#include<iostream>

#include<iomanip>

#define \_CRT\_SECURE\_NO\_WARNINGS

using namespace std;

struct Stack

{

char info;

Stack\* next;

};

struct NumberStack

{

float num;

NumberStack\* next;

};

void Push(Stack\*, char);

char Pop(Stack\*);

void Push(NumberStack\*, float);

float Pop(NumberStack\*);

int my\_strlen(char\*);

int precedence(char);

int isSign(char);

int isParenthesis(char);

char top(Stack\*);

char top(NumberStack\*);

float evaluateToPostfix(Stack\*, char\*);

float evaluate(NumberStack\*, char);

void infixToPostfix(Stack\*, char\*, char\*);

void MemoryCleaning(Stack\*);

int main()

{

Stack\* head = new Stack;

head->next = NULL;

char infix[30];

char postfix[30];

Stack\* signStack = new Stack;

NumberStack\* numberStack = new NumberStack;

int choice, len, postf\_len = 0;

float result;

int isDigit;

while (true)

{

printf\_s("\n 1 - Enter an expression (alphabetic or integer) 2 - Convert to reverse polish notation 3 - Examination else - Exit\n");

scanf("%d", &choice);

switch (choice)

{

case 1:

postf\_len = 0;

memset(postfix, '\0', 30);

printf\_s("Enter an expression: ");

scanf("%s", infix);

isDigit = 1;

len = my\_strlen(infix);

break;

case 2:

signStack->next = NULL;

int i, j;

memset(postfix, '\0', 30);

for (i = 0, j = 0; infix[i] != '\0'; i++)

{

if (isdigit(infix[i]))

{

postfix[postf\_len] = infix[i];

postf\_len++;

j++;

}

else if (isSign(infix[i]))

{

while ((precedence(infix[i])) <= precedence(top(signStack)) && top(signStack) != ')')

{

postfix[postf\_len] = Pop(signStack);

postf\_len++;

j++;

}

Push(signStack, infix[i]);

}

else if (isParenthesis(infix[i]))

{

if (infix[i] == '(')

{

Push(signStack, infix[i]);

}

else

{

while (top(signStack) != '(')

{

postfix[postf\_len] = Pop(signStack);

postf\_len++;

j++;

}

//postfix[j] = '\0';

Pop(signStack);

}

}

else

{

postfix[postf\_len] = infix[i];

isDigit = 0;

postf\_len++;

}

}

while (signStack->next != NULL)

{

postfix[postf\_len] = Pop(signStack);

postf\_len++;

}

postfix[postf\_len] = '\0';

printf\_s("%s\n", postfix);

break;

case 3:

float num;

if (isDigit)

{

result = evaluateToPostfix(head, postfix);

printf\_s("Result is: %4.3f\n", result);

break;

}

for (i = 0; postfix[i] != '\0'; i++)

{

if (isSign(postfix[i]))

{

result = evaluate(numberStack, postfix[i]);

continue;

}

if (isdigit(postfix[i]))

{

Push(numberStack, (float)(postfix[i] - '0'));

continue;

}

printf\_s("Enter value (%c): ", postfix[i]);

getchar();

scanf("%f", &num);

Push(numberStack, num);

}

//result = evaluateToPostfix(head, postfix);

printf\_s("Result is: %4.2f\n", result);

break;

default:

printf\_s("Error: incorrect input!!!");

return 0;

}

}

delete head;

return 0;

}

void Push(Stack\* head, char info)

{

Stack\* newElement = new Stack;

newElement->info = info;

if (head->next == NULL)

newElement->next = NULL;

else

newElement->next = head->next;

head->next = newElement;

}

void Push(NumberStack\* head, float num)

{

NumberStack\* newElement = new NumberStack;

newElement->num = num;

if (head->next == NULL)

newElement->next = NULL;

else

newElement->next = head->next;

head->next = newElement;

}

char Pop(Stack\* head)

{

Stack\* tempNode;

char info;

if (head->next == NULL)

{

printf\_s("Stack is empty");

return 0;

}

else

{

tempNode = head->next;

info = tempNode->info;

head->next = tempNode->next;

delete tempNode;

return info;

}

}

float Pop(NumberStack\* head)

{

NumberStack\* tempNode;

float num;

if (head->next == NULL)

{

printf\_s("Stack is empty");

return 0;

}

else

{

tempNode = head->next;

num = tempNode->num;

head->next = tempNode->next;

delete tempNode;

return num;

}

}

int my\_strlen(char\* infix)

{

int i;

for (i = 0; infix[i] != '\0'; i++);

return i;

}

int precedence(char sign)

{

switch (sign)

{

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case'^':

return 3;

default:

return 0;

}

}

int isSign(char sign)

{

return sign == '+' || sign == '-' || sign == '\*' || sign == '^' || sign == '/';

}

int isParenthesis(char sign)

{

return sign == '(' || sign == ')';

}

char top(Stack\* head)

{

if (head->next == NULL)

{

return '\0';

}

else

{

Stack\* tempNode = head->next;

return tempNode->info;

}

}

float evaluateToPostfix(Stack\* head, char postfix[]) {

Stack\* signStack = new Stack;

signStack->next = NULL;

int i;

for (i = 0; postfix[i] != '\0'; i++)

{

if (isdigit(postfix[i]))

{

Push(signStack, postfix[i] - '0');

}

else if (isSign(postfix[i]))

{

int sign1 = Pop(signStack);

int sign2 = Pop(signStack);

switch (postfix[i])

{

case'+':

Push(signStack, sign2 + sign1);

break;

case'-':

Push(signStack, sign2 - sign1);

break;

case'\*':

Push(signStack, sign2 \* sign1);

break;

case'/':

Push(signStack, sign2 / sign1);

break;

case'^':

Push(signStack, pow(sign2, sign1));

break;

}

}

}

return Pop(signStack);

}

float evaluate(NumberStack\* numberStack, char sign)

{

float num1, num2;

num2 = Pop(numberStack);

num1 = Pop(numberStack);

switch (sign)

{

case'+':

Push(numberStack, num1 + num2);

return num1 + num2;

break;

case'-':

Push(numberStack, num1 - num2);

return num1 - num2;

break;

case'\*':

Push(numberStack, num1 \* num2);

return num1 \* num2;

break;

case'/':

Push(numberStack, num1 / num2);

return num1 / num2;

break;

case'^':

Push(numberStack, pow(num1, num2));

return pow(num1, num2);

break;

}

}

void MemoryCleaning(Stack\* head)

{

while (head->next != NULL)

Pop(head);

}

Вывод: изучила правила формирования постфиксной записи арифметических выражений.

