МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

Отчет

Практическая работа №7

Дисциплина Структуры и алгоритмы обработки данных

Тема. Использование линейных структур данных стека и очереди в алгоритмах Использование стека и очереди в алгоритмах преобразования инфиксной записи арифметических выражений в польскую запись и вычисление значений выражений

Выполнил студент Дамарад Д.В.

Фамилия И.О.

Группа ИКБО-13-21

Номер группы

Задание 1

1. Условие задачи 1:
   Провести преобразование инфиксной записи выражения (столбец 1 таблицы вариантов) в постфиксную нотацию, расписывая процесс по шагам

2. Дано. Арифметическое выражение в инфиксной записи, представленное строкой из n символов, каждый из которых является либо операндом, либо операцией – S=(a+b*c)/(d*e-f)*(g*k+l)

3. Результат. Строка (постфиксная), содержащая постфиксную запись арифметического выражения – S= abc*+de*f-/gk*l+*

4. Алгоритм преобразования инфиксной записи выражения в постфиксную:
   - Сканировать вводимую строку слева направо символ за символом.
   - Если символ является операндом, поместить его в очередь вывода.
   - Если символ является оператором, а стек оператора пуст, вставить оператора в стек оператора.
   - Если стек оператора не пуст, могут быть следующие варианты:
     - Если приоритет сканируемого оператора больше, чем у самого верхнего оператора очереди оператора, поместить этот оператор в стек оператора.
     - Если приоритет отсканированного оператора меньше или равен самому верхнему оператору стека оператора, извлекать операторы из стека оператора и вставлять их в очередь вывода до тех пор, пока не найдется оператор с более низким приоритетом, чем отсканированный символ, после чего вставить отсканированный оператор в стек оператора.
     - Если символ открывает круглую скобку (‘(‘), вставить его в стек оператора.
     - Если символ закрывает круглую скобку (‘)’), вытаскивать операторы из стека оператора и вставлять их в очередь вывода, пока не найдется открывающий скобку (‘(‘).
   - Извлечь все оставшиеся операторы из стека оператора и вставить в очередь вывода.

Элемент выражения	Стек оператора	Очередь вывода
(	(	
a	(	a
+	(+	a
b	(+	ab
*	(+*	ab
c	(+*	abc
)		abc*+
/	/	abc*+
(	/(	abc*+
d	/(	abc*+d
*	/(*	abc*+d
e	/(*	abc*+de
-	/(-	abc*+de*
f	/(-	abc*+de*f
)	/	abc*+de*f-
*	*	abc*+de*f-/
(	*(	abc*+de*f-/
g	*(	abc*+de*f-/g
*	*(*	abc*+de*f-/g
k	*(*	abc*+de*f-/gk
+	*(+	abc*+de*f-/gk*
l	*(+	abc*+de*f-/gk*l
)	*	abc*+de*f-/gk*l+
Конечный результат		abc*+de*f-/gk*l+*

5. Условие задачи 2:
   Представить инфиксную нотацию выражения (столбец 2 таблицы вариантов) (идентификаторы одно символьные) с расстановкой скобок, расписывая процесс по шагам

6. Дано. Арифметическое выражение в постфиксной записи, представленное строкой из n символов, каждый из которых является либо операндом, либо операцией – S= abc*+d/xy*zk/-+m+
   Результат. Строка (инфиксная), содержащая инфиксную запись арифметического выражения – S= ((((a+(b*c))/d)+((x*y)-(z/k)))+m)

7. Алгоритм преобразования постфиксной записи в инфиксную:
   - Сканировать вводимую строку слева направо символ за символом
   - В стек операндов записывается встретившийся операнд
   - Как только попадается оператор, из стека операндов извлекаются последние два операнда, преобразуются в строку вида: (операнд2 оператор операнд1). Полученная конструкция дописывается в стек операндов

8. Таблица стеков:

Элемент выражения	Стек инфиксной формы
a	a
b	ab
c	abc
*	a(b*c)
+	(a+(b*c))
d	(a+(b*c))d
/	((a+(b*c))/d)
x	((a+(b*c))/d)x
y	((a+(b*c))/d)xy
*	((a+(b*c))/d)(x*y)
z	((a+(b*c))/d)(x*y)z
k	((a+(b*c))/d)(x*y)zk
/	((a+(b*c))/d)(x*y)(z/k)
-	((a+(b*c))/d)((x*y)-(z/k))
+	(((a+(b*c))/d)+((x*y)-(z/k)))
m	(((a+(b*c))/d)+((x*y)-(z/k)))m
+	((((a+(b*c))/d)+((x*y)-(z/k))))+m
Конечный результат	((((a+(b*c))/d)+((x*y)-(z/k)))+m)

9. Условие задачи 3:
   Представить префиксную нотацию выражения, полученного в результате выполнения задачи 2, расписывая процесс по шагам.

10. Дано. Арифметическое выражение в инфиксной записи, представленное строкой из n символов, каждый из которых является либо операндом, либо операцией:
    S= ((((a+(b*c))/d)+((x*y)-(z/k)))+m)
    Результат. Строка (префиксная), содержащая префиксную запись арифметического выражения – S= ++/+a*bcd-*xy/zkm.

Алгоритм преобразования инфиксной записи в префиксную:

- Конвертировать инфиксную строку в постфиксный формат
- Сканировать постфиксную строку слева направо символ за символом
- Если символ является операндом, поместить его в стек операнда
- Если символ является оператором, то извлечь два операнда из стека, записать их в виде: оператор операнд2 операнд1. Полученную строку вставить обратно в стек операнда.

11. Таблица стеков:

Элемент выражения	Стек операндов	Полученная строка
a	a	
b	ab	
c	abc	
*	abc	a*bc
+	a*bc	+a*bc
d	+a*bcd	+a*bcd
/	+a*bcd	/+a*bcd
x	/+a*bcdx	/+a*bcdx
y	/+a*bcdxy	/+a*bcdxy
*	/+a*bcdxy	/+a*bcd*xy
z	/+a*bcd*xyz	/+a*bcd*xyz
k	/+a*bcd*xyzk	/+a*bcd*xyzk
/	/+a*bcd*xyzk	/+a*bcd*xy/zk
-	/+a*bcd*xy/zk	/+a*bcd-*xy/zk
+	/+a*bcd-*xy/zk	+/+a*bcd-*xy/zk
m	+/+a*bcd-*xy/zkm	+/+a*bcd-*xy/zkm
+	++/+a*bcd-*xy/zkm	++/+a*bcd-*xy/zkm
Конечный результат		++/+a*bcd-*xy/zkm

12. Условие задания 4:

    Вычислить значение выражения, представленного в столбце 3

13. Дано. Арифметическое выражение в постфиксной записи, представленное строкой из n символов, каждый из которых является либо операндом, либо операцией – S= *-*365+*234
    Результат. Значение вычисленного выражения: S= 130.
14. Алгоритм вычисления префиксного выражения:
    - Сканировать вводимую строку справа налево посимвольно
    - В стек операндов записывается встретившийся операнд
    - Как только попадается оператор, из стека операндов извлекаются последние два операнда, преобразуются в строку вида: (операнд2 оператор операнд1). Полученное выражение вычисляется и записывается в стек операнда
15. Таблица стеков:

Элемент выражения	Стек операндов	Выражение	Результат
4	4		
3	43		
2	432		
*	432	2*3	6
+	4(6)	6+4	10
5	(10)5		
6	(10)56		
3	(10)563		
*	(10)563	3*6	18
-	(10)5(18)	18-5	13
*	(10)(13)	13*10	130
Конечный результат			130

Задание 2

1. Условие задачи:

   Выполнить программную реализацию задачи варианта. Дано арифметическое выражение в постфиксной или префиксной форме, представленной в строковом формате. Операнды однозначные числа.

№	Форма выражения	Структура реализации стека или очереди	Задача
5	Инфиксная	Массив	Вычислить значение выражения

2. Постановка задачи:

   1) Определить форму записи выражения.
   2) Разработать АТД задачи.
   3) Реализовать структуру АТД задачи на стеке или очереди в зависимости от формы выражения варианта. Реализацию стека или очереди выполнить в соответствии со структурой, определенной в варианте. Операции над стеком и очередью реализовать как отдельные функции.
   4) Провести тестирование разработанного приложения.

3. Постановка задачи:
   Дано. Выражение, записанное в инфиксной форме.
   Результат. Вычисленное значение выражения.

4. АТД задачи:
   - Строковое выражение читается слева направо символ за символом.
   - Если символ является операндом, записать его в очередь операндов
   - Если символ является оператором, то сохраняем его в специальную переменную
   - Когда встретился очередной операнд, и кол-во операндов в очереди стало равнятся двум, вытаскиваем из очереди операнды и, в зависимости от операнда, проводим над ними операцию (или сложения или умножения или деления или вычитания). Полученное число записываем обратно в очередь

5. Код реализации АТД:

```
int SolutionOfInfixForm(string infix, queue& mystack)
{
    char operation = '\n'; // переменная, хранящяя знак операции над числами
    for (int i = 0; i < infix.size(); i++)
    {
        if ('0' <= infix[i] and infix[i] <= '9')
        {
            mystack.push_back(infix[i] - '0');
            if (mystack.size == 2) // если кол-во чисел в очереди уже 2
            {
                if (operation == '+')
                mystack.push_back(mystack.pop() + mystack.pop());
                if (operation == '-')
                mystack.push_back(mystack.pop() - mystack.pop());
                if (operation == '/')
                mystack.push_back(mystack.pop() / mystack.pop());
                if (operation == '*')
                mystack.push_back(mystack.pop() * mystack.pop());
            }
        }
        else
        {
            if(infix[i]!='(' and infix[i] != ')') // проверяем на то, что не сохраняем скобки
                operation = infix[i]; // запоминаем операцию
        }
    }
    return mystack.pop(); // в конце возвращаем полученное число из стека
}
```

6. Код реализации программы:

```
#include <iostream>
#include <string.h>
using namespace std;

const int MAX = 2; // максимальное кол-во чисел в очереди - 2

struct queue {
private:
	int array[MAX] = {};
public:
	queue() { // конструктор
		size = 0;
	}
	~queue() { // деструктор
		this->pop();
		this->pop();
	}
	void push_back(int x); // метод заноса элемента в очереди
	int pop(); // метод вытаскивания последнего элемента из очереди
	int size; // поле размера стека
};

int queue::pop() { // функция 'вытягивания' последнего элемента очереди
	if (size == 0) return 0; // проверка на размер очереди
	int data = array[0]; // запоминаем значение последнего элемента
	array[0] = array[1];
	array[1] = 0;
	size--; // уменьшаем размер очереди
	return data; // возвращаем сохраненное значение
}

void queue::push_back(int x) { // обычный занос в очереди
	array[size] = x; // присваиваем последней ячейке (по размеру) значение
x
	size++; // увеличиваем размер стэка
}

int SolutionOfInfixForm(string infix, queue& mystack) {
	char operation = '\n'; // переменная, хранящяя знак операции над числами
	for (int i = 0; i < infix.size(); i++) {
		if ('0' <= infix[i] and infix[i] <= '9') {
			mystack.push_back(infix[i] - '0');
```

```
                if (mystack.size == 2) { // если кол-во чисел в очереди уже 2

                    if (operation == '+')
                        mystack.push_back(mystack.pop() +
mystack.pop());
                    if (operation == '-')
                        mystack.push_back(mystack.pop() -
mystack.pop());
                    if (operation == '/')
                        mystack.push_back(mystack.pop() /
mystack.pop());
                    if (operation == '*')
                        mystack.push_back(mystack.pop() *
mystack.pop());
                }
            }
            else {
                if (infix[i] != '(' and infix[i] != ')') { // проверяем на то, что не
сохраняем скобки
                    operation = infix[i]; // запоминаем операцию
                }
            }
        }
        return mystack.pop(); // в конце возвращаем полученное число из стека
}

int main() {
        setlocale(LC_ALL, "");
        queue mystack;
        string infix;
        cout << "Введите выражение в инфиксной форме:  ";
        cin >> infix;
        cout << "Значение выражения:  " << SolutionOfInfixForm(infix, mystack)
<< endl;
}
```

7. Результаты тестирования программы: