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 Черномуров С. А.
Фамилия И.О.

Группа ИКБО-13-21
Номер группы

Москва 2022

Задание 1

Вариант №23

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=x+y*z/d-a*b-c/(j-k).
   Результат. Строка (постфиксная), содержащая постфиксную запись арифметического выражения – S=xyz*d/+ab*-cjk-/-.
3. 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.
4. Таблица стеков и очередей:

Элемент выражения	Стек оператора	Очередь вывода
x		x
+	+	x
y	+	xy
*	+*	xy
z	+*	xyz
/	+/	xyz*
d	+/	xyz*d
-	-	xyz*d/+
a	-	xyz*d/+a
*	-*	xyz*d/+a
b	-*	xyz*d/+ab
-	-	xyz*d/+ab*-
c	-	xyz*d/+ab*-c
/	-/	xyz*d/+ab*-c
(	-/(	xyz*d/+ab*-c
j	-/(	xyz*d/+ab*-cj
-	-/(-	xyz*d/+ab*-cj
k	-/(-	xyz*d/+ab*-cjk
)	-/	xyz*d/+ab*-cjk-
конец строки		xyz*d/+ab*-cjk-/-

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=afbc*-zx-/y+ar*k-/+.
   Результат. Строка (инфиксная), содержащая инфиксную запись арифметического выражения – S=(a+((((f-(b*c))/(z-x))+y)/((a*r)-k))).
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 операндов	Полученная строка
a	a	
f	af	
b	afb	
c	afbc	
*	af (b*c)	(b*c)
-	a (f-(b*c))	(f-(b*c))
z	a (f-(b*c)) z	(f-(b*c))
x	a (f-(b*c)) zx	(f-(b*c))
-	a (f-(b*c)) (z-x)	(f-(b*c))(z-x)
/	a ((f-(b*c))/(z-x))	(f-(b*c))/(z-x)
y	a ((f-(b*c))/(z-x)) y	(f-(b*c))/(z-x)
+	a (((f-(b*c))/(z-x))+y)	(((f-(b*c))/(z-x))+y)
a	a (((f-(b*c))/(z-x))+y) a	(((f-(b*c))/(z-x))+y)
r	a (((f-(b*c))/(z-x))+y) ar	(((f-(b*c))/(z-x))+y)
*	a (((f-(b*c))/(z-x))+y) (a*r)	(((f-(b*c))/(z-x))+y) (a*r)
k	a (((f-(b*c))/(z-x))+y) (a*r) k	(((f-(b*c))/(z-x))+y) (a*r)
-	a (((f-(b*c))/(z-x))+y) ((a*r)-k)	(((f-(b*c))/(z-x))+y) ((a*r)-k)
/	a ((((f-(b*c))/(z-x))+y)/((a*r)-k))	((((f-(b*c))/(z-x))+y)/((a*r)-k))
+	a+((((f-(b*c))/(z-x))+y)/((a*r)-k))	(a+((((f-(b*c))/(z-x))+y)/((a*r)-k)))

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+((((f-(b*c))/(z-x))+y)/((a*r)-k).
    Результат. Строка (префиксная), содержащая префиксную запись арифметического выражения – S=+a/+/-f*bc-zxy-*ark.
11. 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.
12. Таблица стеков:

Элемент выражения	Стек операндов	Полученная строка
a	a	
f	af	
b	afb	

c	afbc	
*	af *bc	*bc
-	a -f*bc	-f*bc
z	a -f*bc z	-f*bc
x	a -f*bc zx	-f*bc
-	a -f*bc -zx	-f*bc -zx
/	a /-f*bc-zx	/-f*bc-zx
y	a /-f*bc-zx y	/-f*bc-zx
+	a +/-f*bc-zxy	+/-f*bc-zxy
a	a +/-f*bc-zxy a	+/-f*bc-zxy
r	a +/-f*bc-zxy ar	+/-f*bc-zxy
*	a +/-f*bc-zxy *ar	+/-f*bc-zxy *ar
k	a +/-f*bc-zxy *ar k	+/-f*bc-zxy *ar
-	a +/-f*bc-zxy -*ark	+/-f*bc-zxy -*ark
/	a /+/-f*bc-zxy-*ark	/+/-f*bc-zxy-*ark
+	+a/+/-f*bc-zxy-*ark	+a/+/-f*bc-zxy-*ark

13. Условие задания 4:
    Вычислить значение выражения, представленного в столбце 3
14. Дано. Арифметическое выражение в постфиксной записи, представленное строкой из n символов, каждый из которых является либо операндом, либо операцией – S=2 3 4 +2*/2 3 1 ^+*.
    Результат. Значение вычисленного выражения S=0.7142.
15. Алгоритм вычисления постфиксного выражения:
    - 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). Полученное выражение вычисляется и записывается в стек операнда.
16. Таблица стеков:

Элемент выражения	Стек операндов	Полученная строка
2	2	
3	23	
4	234	
+	2 7	3+4=7
2	2 7 2	
*	2 14	7*2=14
/	1/7	2/14=1/7
2	1/7 2	
3	1/7 2 3	
1	1/7 2 3 1	
^	1/7 2 3	3^1=3
+	1/7 5	2+3=5
*	5/7	1/7*5=5/7=0.7142

Задание 2
Вариант №23

1. Условие задания:
   Выполнить программную реализацию задачи варианта. Дано арифметическое выражение в форме, указанной в варианте, представленное в строковом формате. Операнды однозначные числа.

23	Префиксная	Массив	Вычислить значение выражения

2. Требования к выполнению задачи:
   1) Определить форму записи выражения (префиксная или постфиксная).
   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 префиксной форме.
   Результат. Вычисленное значение выражения.
4. АТД задачи:
   - Данное строковое выражение разворачивается(в случае программы просто читается справа налево)
   - Строковое выражение читается слева направо символ за символом
   - Если символ является операндом, записать его в стек операндов
   - Если символ является оператором, из стека извлекаются два операнда, вычисленное выражение: операнд1 оператор операнд2 записывается в стек операндов
5. Код реализации АТД:
   Предусловие. Строковое выражение prefix, записанное в префиксной форме
   Постусловие. result – вычисленное значение выражения.

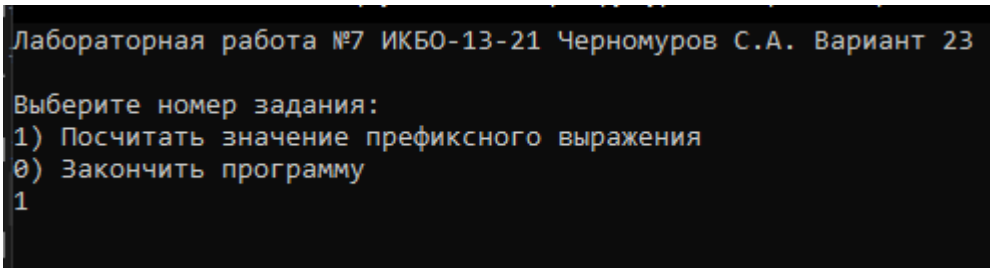
```
int solve_prefix(string prefix);
```

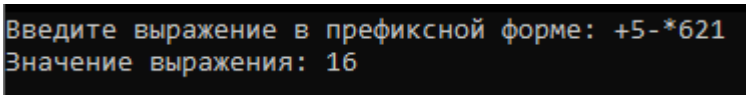
Тест функции:

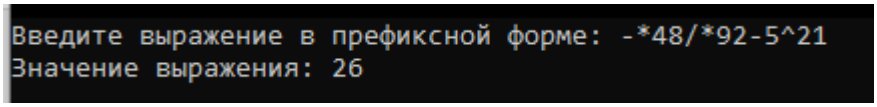
Номер теста	Исходные данные	Ожидаемый результат
1	+5-*621	16
2	-*48/*92-5^21	26

```
int solve_prefix(string prefix) {
	Stack operandStack("",prefix.size());
	for (int i = prefix.size() - 1; i >= 0; i--) {
		if (isdigit(prefix[i])) operandStack.AddEl(prefix[i] - '0');
		else {
			int operand1;
			operandStack.TakeEl(operand1);

			int operand2;
			operandStack.TakeEl(operand2);

			if (prefix[i] == '+') operandStack.AddEl(operand1 + operand2);
			else
				if (prefix[i] == '-') operandStack.AddEl(operand1 -
operand2);
				else
					if (prefix[i] == '*') operandStack.AddEl(operand1 *
operand2);
					else
						if (prefix[i] == '/')
operandStack.AddEl(operand1 / operand2);
						else
							if (prefix[i] == '^')
operandStack.AddEl(pow(operand1, operand2));
		}
	}
	int result;
	operandStack.TakeEl(result);
	return result;
}
```

Тестирование программы







Код программы на языке C++

Файл functions.cpp (описано тело функции и методы класса Stack)

```cpp
#include "functions.h"
#include <cmath>

int solve_prefix(string prefix) {
	Stack operandStack("",prefix.size()+1);
	for (int i = prefix.size() - 1; i >= 0; i--) {
		if (isdigit(prefix[i])) operandStack.AddEl(prefix[i] - '0');
		else {
			int operand1;
			operandStack.TakeEl(operand1);

			int operand2;
			operandStack.TakeEl(operand2);
			if (prefix[i] == '+') operandStack.AddEl(operand1 + operand2);
			else
				if (prefix[i] == '-') operandStack.AddEl(operand1 -
operand2);
				else
					if (prefix[i] == '*') operandStack.AddEl(operand1 *
operand2);
					else
						if (prefix[i] == '/')
operandStack.AddEl(operand1 / operand2);
						else
							if (prefix[i] == '^')
operandStack.AddEl(pow(operand1, operand2));
		}
	}
	int result;
	operandStack.TakeEl(result);
	return result;
}

Stack::Stack(string _name, int _size) {
	name = _name; // присвоение имени стека
	size = _size; // присвоение размера стека
	a = new int[size]; // выделение памяти для массива
	curlen = -1; // инициализация текущей длины -1
}

int Stack::StackSize() { // возврат размера стека
	return size;
}

int Stack::CurrentLength() { // возврат текущей длины стека
	return curlen + 1;
}

string Stack::StackName() { // возврат имени стека
	return name;
}

bool Stack::AddEl(int _el) {
	curlen++; // инкрементирование текущей длины
	if (curlen <= size - 1) { a[curlen] = _el; return true; } // инициализация
элемента стека
	else
		return false;
}

bool Stack::TakeEl(int& m) {
	if (CurrentLength() > 0) { // извлечение элемента стека
		m = a[CurrentLength() - 1];
		curlen--; // декрементирование текущей длины стека
		return true;
```

```
	}
	else return false;

}
```

Файл functions.h (описан класс Stack)

```
#pragma once
#include <iostream>
#include <string>
using namespace std;

class Stack {
private:
	string name;
	int size;
	int curlen;
	int* a;
public:

	Stack(string _name, int _size);
	bool AddEl(int _el);
	bool TakeEl(int& m);
	string StackName();
	int StackSize();
	int CurrentLength();
};



int solve_prefix(string prefix);
```

Файл main.cpp (основной алгоритм программы)

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

int main() {
	setlocale(LC_ALL, "");

	cout << "Лабораторная работа №7 ИКБО-13-21 Черномуров С.А. Вариант 23" << endl
<< endl;
	cout << "Выберите номер задания:\n1) Посчитать значение префиксного
выражения\n0) Закончить программу\n";

	int choice1;

	do {
		cin >> choice1;

		if (choice1 != 1 && choice1 != 0) cout << "Введено неверное значение,
попробуйте снова.\n";
	} while (choice1 != 1 && choice1 != 0);

	system("cls");

	switch (choice1) {
```

```
        case 1: {
                cout << "Введите выражение в префиксной форме: ";
                string expression;
                cin.ignore(32767, '\n');
                getline(cin, expression);
                cout << "Значение выражения: " << solve_prefix(expression)<<"\n\n";
                break; }

        case 0:
                return 0;
        }
        main();
}
```

Вывод

В ходе выполнения работы были получены знания и навыки по реализации структуры стек и очередь, получены умения и навыки по выполнению операций на структурах стек и очередь, получены знания, умения по представлению арифметических выражений в польской записи, а также выполнено задание в соответствии с персональным вариантом (23)