МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное
учреждениевысшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

ОТЧЕТ
ПО ПРАКТИЧЕСКОЙ РАБОТЕ №3

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

по дисциплине

«СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ»

Выполнил студент Иолович Е.А.

группа ИНБО-03-22

Москва 2023

СОДЕРЖАНИЕ

1 ЗАДАНИЕ 1

1.1 Задача 1

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

В соответствии с индивидуальным вариантом (10 вариант):

S = c + (d * e – a / b) ^ f / g / k

№	Выражение	Стэк	Постфиксное выражение
1	c	(	c
2	+	(+	c
3	d	(+(	cd
4	*	(+(*	cd
5	e	(+(*	cde
6	-	(+(-	cde*
7	a	(+(-	cde*a
8	/	(+(-/	cde*a
9	b	(+(-/	cde*ab
10	)	(+	cde*ab/-
11	^	(+ ^	cde*ab/-
12	f	(+ ^	cde*ab/-f
13	/	(+/	cde*ab/-f^
14	g	(+/	cde*ab/-f^g
15	/	(+/	cde*ab/-f^g/
16	k	(+/	cde*ab/-f^g/k
17	)	-	cde*ab/-f^g/k/+

Получим: **cde*ab/-f^g/k/+**

1.2 Задача 2

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

В соответствии с индивидуальным вариантом (10 вариант):

S = m n k p * + x - / z y ^ *

№ операции	Элемент выражения	Состояние строки	Состояние стэка

1	m	m n k p * + x - / z y ^ *	m
2	n	k p * + x - / z y ^ *	mn
3	k	p * + x - / z y ^ *	mnk
4	p	* + x - / z y ^ *	mnkp
5	*	+ x - / z y ^ *	mn(k*p)
6	+	x - / z y ^ *	m(n+k*p)
7	x	- / z y ^ *	m(n+k*px)
8	-	/ z y ^ *	m(n+k*p-x)
9	/	z y ^ *	m/(n+k*p-x)
10	z	y ^ *	m/(n+k*p-x)z
11	y	^ *	m/(n+k*p-x)zy
12	^	*	m/(n+k*p-x)z^y
13	*	-	**m/(n+k*p-x)*z^y**

1.3 Задача 3

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

Из предыдущего пункта имеем:

S = m / (n + k * p - x) * z ^ y

№	Элемент выражения	Постфиксное выражение	Состояние стэка
1		m/(n+k*p-x)*z^y	
2	m	/(n+k*p-x)*z^y	m
3	/	(n+k*p-x)*z^y	/m
4	(	n+k*p-x)*z^y	/m
5	n	+k*p-x)*z^y	/mn
6	+	k*p-x)*z^y	/m+n
7	k	*p-x)*z^y	/m+kn
8	*	p-x)*z^y	/m+*kn
9	p	-x)*z^y	/m+*kpn
10	-	x)*z^y	/m-+*kpn
11	x	)*z^y	/m-+*kpnx
12	)	*z^y	/m-+*kpnx
13	*	z^y	*/m-+*kpnx
14	z	^y	*/m-+*kpnxz
15	^	y	*/m-+*kpnx^z
16	y	-	***/m-+*kpnx^zy**

1.4 Задача 4

Вычислить значение выражения, представленного в столбце 3, выполняя вычисление по образцу.

В соответствии с индивидуальным вариантом (10 вариант):

S = 3 7 2 * + 3 6 5 * - / 2 3 4 * + *

№ операции	Элемент выражения	Состояние строки	Состояние стэка
1	3	3 7 2 * + 3 6 5 * - / 2 3 4 * +*	3
2	7	7 2 * + 3 6 5 * - / 2 3 4 * + *	3 7
3	2	2 * + 3 6 5 * - / 2 3 4 * + *	3 7 2
4	*	* + 3 6 5 * - / 2 3 4 * + *	3 14
5	+	+ 3 6 5 * - / 2 3 4 * + *	17
6	3	3 6 5 * - / 2 3 4 * + *	17 3
7	6	6 5 * - / 2 3 4 * + *	17 3 6
8	5	5 * - / 2 3 4 * + *	17 3 6 5
9	*	* - / 2 3 4 * + *	17 3 30
10	-	- / 2 3 4 * + *	17 -27
11	/	/ 2 3 4 * + *	-17/27
12	2	2 3 4 * + *	-17/27 2
13	3	3 4 * + *	-17/27 2 3
14	4	4 * + *	-17/27 2 3 4
15	*	* + *	-17/27 2 12
16	+	+ *	-17/27 14
17	*	*	-238,27

Получим: - 238/27

2 ЗАДАНИЕ 2

2.1 Условие задачи

Дано арифметическое выражение в форме, указанной в варианте, представленное в строковом формате. Операнды однозначные числа.

1. Определить форму записи выражения (префиксная или постфиксная).
2. Разработать АТД (абстрактный тип данных) 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.

2.2 Постановка задачи

В соответствии с индивидуальным вариантом (10 вариант):

Форма выражения	**Структура реализации стека или очереди**	**Задача**
Инфиксная	Список	Вычислить значение выражения

2.3 АТД задачи

Сформировать две структуры: для информационных узлов списка, на которых основан стек по заданию и сама структура стека.

2.4 Код реализации АТД

```
3  //структура для элементов стека.
4  struct Node
5  {
6      char data;
7      Node* next;
8  };
```

```
9
10 //структура для стека.
11 struct Stack
12 {
13     Node* top;
14     int size;
15 };
```

2.5 Код основной программы

```
16 #include <iostream>
17 #include <string>
18
19 using namespace std;
20
21 //структура для элементов стека.
22 struct Node
23 {
24     char data;
25     Node* next;
26 };
27
28 //структура для стека.
29 struct Stack
30 {
31     Node* top;
32     int size;
33 };
34
35 //функция для инициализации стека.
36 void initStack(Stack* s)
37 {
38     s->top = NULL;
39     s->size = 0;
40 }
41
42 //функция для проверки, пуст ли стек.
43 bool isEmpty(Stack* s)
44 {
45     return s->top == NULL;
46 }
47
48 //функция для добавления элемента в стек.
49 void push(Stack* s, char data)
50 {
51     Node* newNode = new Node;
52     newNode->data = data;
53     newNode->next = s->top;
54     s->top = newNode;
55     s->size++;
```

```
56 }
57
58 //функция для удаления элемента из стека.
59 char pop(Stack* s)
60 {
61     if (isEmpty(s))
62     {
63         cout << "Стек пуст" << endl;
64         return '\0';
65     }
66
67     Node* temp = s->top;
68     char data = temp->data;
69     s->top = temp->next;
70     delete temp;
71     s->size--;
72     return data;
73 }
74
75 //функция для получения верхнего элемента стека без его удаления.
76 char peek(Stack* s)
77 {
78     if (isEmpty(s))
79     {
80         cout << "Стек пуст" << endl;
81         return '\0';
82     }
83
84     return s->top->data;
85 }
86
87 //функция для определения приоритета оператора.
88 int precedence(char op)
89 {
90     if (op == '+' || op == '-')
91     {
92         return 1;
93     }
94     if (op == '*' || op == '/')
95     {
96         return 2;
97     }
98     return 0;
99 }
100
101//функция для вычисления инфиксной записи.
102int evaluateInfixExpression(string expression)
103{
104     Stack operands;  //инициализация стеков для операндов.
105     initStack(&operands);
```

```
106
107     Stack operators; //инициализация стеков для операторов.
108     initStack(&operators);
109
110     for (int i = 0; i < expression.length(); i++) //проход по каждому
    символу в строке.
111     {
112         char c = expression[i];
113
114         if (c == ' ') //если символ пробел, то пропустить его.
115         {
116             continue;
117         }
118
119         if (isdigit(c)) //если символ цифра, то прочитать все последующие
    цифры, чтобы получить операнд и добавить его в стек операндов.
120         {
121             int num = 0;
122             while (isdigit(c))
123             {
124                 num = c - '0'; //учитывает однознач. числа.
125                 i++;
126                 c = expression[i];
127             }
128             i--;
129
130             push(&operands, num);
131         }
132         else if (c == '(') //если символ откр. скобочка, то добавить его в
    стек операторов.
133         {
134             push(&operators, c);
135         }
136         else if (c == ')')
137         {
138             while (peek(&operators) != '(') //если символ закр. скобочка,
    то вытаскивать из элементы стека операторов, пока не встретится откр.
    скобочка.
139             { //для двух последних операндов выполнять арифметическое дей-
    ствие и засунуть обратно в стек с операторами.
140                 int operand2 = pop(&operands);
141                 int operand1 = pop(&operands);
142                 char op = pop(&operators);
143
144                 switch (op)
145                 {
146                 case '+':
147                     push(&operands, operand1 + operand2);
148                     break;
149                 case '-':
150                     push(&operands, operand1 - operand2);
151                     break;
```

```
152                 case '*':
153                     push(&operands, operand1 * operand2);
154                     break;
155                 case '/':
156                     push(&operands, operand1 / operand2);
157                     break;
158                 }
159             }
160             pop(&operators);
161         }
162         else
163         {
164             while (!isEmpty(&operators) && precedence(c) <= prece-
    dence(peek(&operators)))
165             {
166                 int operand2 = pop(&operands);
167                 int operand1 = pop(&operands);
168                 char op = pop(&operators);
169                 //если оператор является одним из арифметических знаков,
    то выталкивать элементы из стека операторов, пока не будет найден
170                 //оператор с равным\наименьшим приоритетом. на каждый опе-
    ратор приходится два операнда, которые нужно извлекать из стека операндов.
171                 //нужно выполнить операцию между двумя операндами и ре-
    зультат обратно добавить в стек операндов.
172                 switch (op)
173                 {
174                 case '+':
175                     push(&operands, operand1 + operand2);
176                     break;
177                 case '-':
178                     push(&operands, operand1 - operand2);
179                     break;
180                 case '*':
181                     push(&operands, operand1 * operand2);
182                     break;
183                 case '/':
184                     push(&operands, operand1 / operand2);
185                     break;
186                 }
187             }
188             push(&operators, c); //добавление текущего оператора в стек
    операторов.
189         }
190     }
191
192     while (!isEmpty(&operators)) //после анализа всех операндов и опера-
    торов, доставать из стеков операнды и операторы, пока не останется
193         //только один операнд в стеке операндов. этот операнд ввернуть как
    значение выражения.
194     {
195         int operand2 = pop(&operands);
196         int operand1 = pop(&operands);
```

```
197         char op = pop(&operators);
198
199         switch (op)
200         {
201         case '+':
202             push(&operands, operand1 + operand2);
203             break;
204         case '-':
205             push(&operands, operand1 - operand2);
206             break;
207         case '*':
208             push(&operands, operand1 * operand2);
209             break;
210         case '/':
211             push(&operands, operand1 / operand2);
212             break;
213         }
214     }
215
216     return pop(&operands);
217 }
218
219 int main() {
220     setlocale(LC_ALL, "Russian");
221     string expression;
222     cout << "Введите выражение: ";
223     getline(cin, expression);
224     int result = evaluateInfixExpression(expression);
225     cout << "Результат: " << result << endl;
226 }
```

2.6 Результат тестирования программы

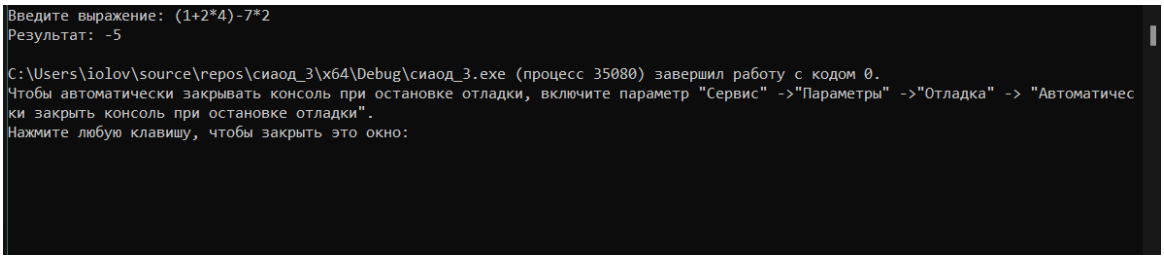

Рисунок 1 – результат тестирования программы

3 ВЫВОДЫ

Стек — это область оперативной памяти, которая создаётся для каждого потока. Он работает в порядке LIFO (Last In, First Out), то есть последний добавленный в стек кусок памяти будет первым в очереди на вывод из стека. Из-за такой природы стека управление памятью оказывается весьма логичным и простым для выполнения разных действий. Тем не менее, у такой строгой формы управления есть и недостатки. Размер стека — это фиксированная величина, и превышение лимита выделенной на стеке памяти приведёт к переполнению стека. Кроме того, переменные, расположенные на стеке, всегда являются локальными. В итоге стек позволяет управлять памятью наиболее эффективным образом.

4 СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Материалы по дисциплине (Сорокин А.В.).

2. Скворцова Л.А., Гусев К.В., Трушин С.М., Филатов А.С. Учебно-методическое пособие СиАОД.