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 Смольников А.Б.
Фамилия И.О.

Группа ИКБО-13-21
Номер группы

Москва 2022

Задание 1

Вариант №20

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)*p*(m+n^k).
3. Результат. Строка (постфиксная), содержащая постфиксную запись арифметического выражения.
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.
5. Таблица стеков и очередей:

Элемент выражения	Стек оператора	Очередь вывода
x		x
/	/	x
(	/(	x
y	/(	xy
+	/(+	xy
z	/(+	xyz
)	/	xyz+
*	*	xyz+/
p	*	xyz+/p
*	*	xyz+/p*
(	*(	xyz+/p*
m	*(	xyz+/p*m
+	*(+	xyz+/p*m
n	*(+	xyz+/p*mn
^	*(+^	xyz+/p*mn
k	*(+^	xyz+/p*mnk
)	*	xyz+/p*mn*+
конец строки		**xyz+/p*mn*+**

6. Условие задачи 2:
   Представить инфиксную нотацию выражения (столбец 2 таблицы вариантов) (идентификаторы одно символьные) с расстановкой скобок, расписывая процесс по шагам
7. Дано. Арифметическое выражение в постфиксной записи, представленное строкой из n символов, каждый из которых является либо операндом, либо операцией – a d e -*f g+c/+.
   Результат. Строка (инфиксная), содержащая инфиксную запись арифметического выражения
8. 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

9. Таблица стеков:

Элемент выражения	Стек операндов	Полученная строка
a	a	
d	ad	
e	ade	
-	a(d-e)	(d-e)
*	(a*(d-e))	(a*(d-e))
f	(a*(d-e))f	(a*(d-e))
g	(a*(d-e))fg	(a*(d-e))
+	(a*(d-e))(f+g)	(a*(d-e))(f+g)
c	(a*(d-e))(f+g)c	(a*(d-e))(f+g)
/	(a*(d-e))((f+g)/c)	(a*(d-e))((f+g)/c)
+	((a*(d-e))+((f+g)/c))	((a*(d-e))+((f+g)/c))
Конец строки		**((a*(d-e))+((f+g)/c))**

10. Условие задачи 3:
    Представить префиксную нотацию выражения, полученного в результате выполнения задачи 2, расписывая процесс по шагам.
11. Дано. Арифметическое выражение в инфиксной записи, представленное строкой из n символов, каждый из которых является либо операндом, либо операцией – **((a*(d-e))+((f+g)/c))**
    Результат. Строка (префиксная), содержащая префиксную запись арифметического выражения.
12. Алгоритм преобразования инфиксной записи в префиксную:
    - Конвертировать инфиксную строку в постфиксный формат согласно алгоритму задачи 1
    - 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.

13. Таблица стеков:

Элемент выражения	Стек операндов	Полученная строка
a	a	
d	ad	
e	ade	
-	a-de	-de
*	*a-de	*a-de
f	*a-def	*a-de
g	*a-defg	*a-de
+	*a-de+fg	*a-de+fg
c	*a-de+fgc	*a-de+fg
/	*a-de/+fgc	*a-de/+fgc
+	+*a-de/+fgc	+*a-de/+fgc
Конец строки		**+*a-de/+fgc**

14. Условие задания 4:
    Вычислить значение выражения, представленного в столбце 3
15. Дано. Арифметическое выражение в постфиксной записи, представленное строкой из n символов, каждый из которых является либо операндом, либо операцией – S=/*2+6*3 2^-7+4 1 3
    Результат. Значение вычисленного выражения
16. Алгоритм вычисления постфиксного выражения:
    - Развернем строку, получим S1=314+7-^23*6+2*/
    - Сканировать S1 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: (операнд1 оператор операнд2). Полученное выражение вычисляется и записывается в стек операнда.
17. Таблица стеков:

Элемент выражения	Стек операндов	Полученная строка
3	3	
1	3 1	
4	3 1 4	
+	3 5	4+1=5
7	3 5 7	
-	3 2	7 – 5 = 2
^	8	2^3 = 8
2	8 2	

3	8 2 3	
*	8 6	3*2=6
6	8 6 6	
+	8 12	6+6=12
2	8 12 2	
*	8 24	2*12=24
/	**3**	24/8=3

Задание 2
Вариант №20

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

20	Инфиксная	Список	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 инфиксной форме
   Результат. Вычисленное значение выражения.
4. АТД задачи:
   - Данное строковое выражение читается справа налево
   - Строковое выражение читается слева направо символ за символом
   - Если символ является операндом, записать его в стек операндов
   - Если символ является оператором, приоритет которого выше приоритета последнего оператора, из стека

извлекаются два операнда, вычисленное выражение: операнд1 оператор операнд2 записывается в стек операндов.

- Открывающаяся скобка всегда записывается в стек операторов, а закрывающаяся скобка «выталкивает» все операторы (выполняются арифметические действия) до открывающейся скобки. Последняя извлекается за ненадобностью.

Таким образом, для реализации данной задачи подходит структура данных стек, реализованная на однонаправленном списке, принцип LIFO(last-in-first-out) вписывается в условие задачи

5. Код реализации АТД:
   Предусловие. Строковое выражение infixForm
   Постусловие. result – вычисленное значение выражения.

```
1.  int calculateInfixForm(string infixForm);
```

Тест функции:

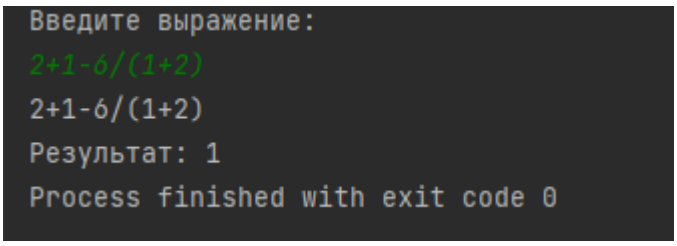
Номер теста	Исходные данные	Ожидаемый результат
1	2+3	5
2	2+1-6/(1+2)	1
3	1+1*9+(2^4-6^2)	-10
4	2^2^2	16

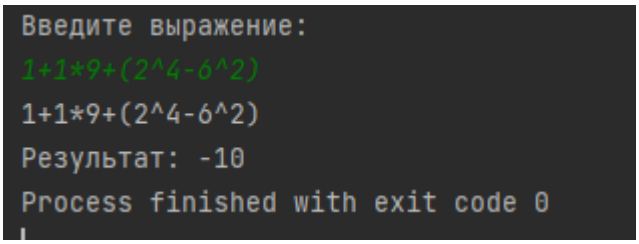
Тестирование программы

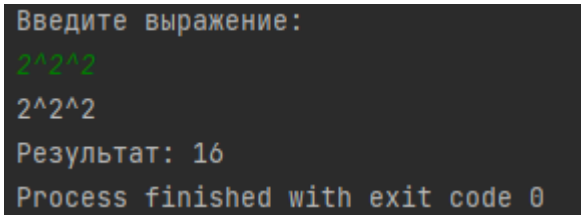
```
D:\Clion\Siaod7\cmake-build-debug\Siaod7.exe
Active code page: 65001
Смольников Алексей. Практическая работа 7. Вариант 20

Подсчитать значение инфиксного выражения, реазизуя стек на однонаправленном списке
1 - ввод выражения, 0 - выход
1
1
```

```
Введите выражение:
2+3
2+3
Результат: 5
Process finished with exit code 0
```







Код программы на языке C++

Файл listStack.h (описана структура данных ListStack)

```
1.  #ifndef SIAOD7_LISTSTACK_H
2.  #define SIAOD7_LISTSTACK_H
3.
4.  #include <string>
5.  using namespace std;
6.
7.  //Стек с использованием списка
8.  template<typename T>
9.  class ListStack {
10. private:
11.     struct Node {
12.         T data;
13.         Node *next;
14.     };
15.     Node *head;
16.     string name;
17.     int size=0;
18. public:
19.     ListStack() {
20.         head = nullptr;
21.     }
22.     ~ListStack() {
23.         while (head != nullptr) {
24.             Node *tmp = head;
25.             head = head->next;
26.             delete tmp;
27.         }
28.     }
29.     bool AddEl(T data){
30.         Node *tmp = new Node;
31.         tmp->data = data;
32.         tmp->next = head;
33.         head = tmp;
34.         size++;
35.         return true;
36.     }
```

```
37.     //take element from end of stack
38.     bool TakeEl(T& m){
39.         if (head == nullptr) {
40.             return false;
41.         }
42.         Node *tmp = head;
43.         m = tmp->data;
44.         head = head->next;
45.         delete tmp;
46.         size--;
47.         return true;
48.     }
49.
50.     bool isEmpty() {
51.         return head == nullptr;
52.     }
53.     string StackName(){
54.         return name;
55.     }
56.     int CurrentLength(){
57.         return size;
58.     }
59. };
60.
61. #endif //SIAOD7_LISTSTACK_H
```

Файл main.cpp (основной алгоритм программы)

```
1.  #include <iostream>
2.  #include <map>
3.  #include <string>
4.  #include "listStack.h"
5.  #include <cmath>
6.  using namespace std;
7.
8.  bool isOperator(char c)
9.  {
10.     if (c == '+' || c == '-' || c == '*' || c == '/' || c == '^'|| c == '(' || c == ')')
11.         return true;
12.     else
13.         return false;
14. }
15. bool isOperand(char c)
16. {
17.     if (c >= '0' && c <= '9')
18.         return true;
19.     else
20.         return false;
21. }
22.
23. int doOper(int a, char oper, int b){
24.     switch(oper){
25.         case '+':
26.             return a+b;
27.         case '-':
28.             return a-b;
29.         case '*':
30.             return a*b;
31.         case '/':
32.             return a/b;
33.         case '^':
34.             return pow(a,b);
35.         default:
36.             return 0;
37.     };
38.
39. }
40.
41.
42. int calculateInfixForm(string infixForm){
43.     map<char,int> precedence;
```

```
    precedence['+'] = 3;
    precedence['-'] = 3;
    precedence['*'] = 2;
    precedence['/'] = 2;
    precedence['^'] = 1;
    precedence['('] = -1;
    //precedence[')'] = 0;
    ListStack<char> operators;
    ListStack<int> operands;
    for(auto c:infixForm){
        if(isOperand(c)){ //если просто число
            operands.AddEl((int)c - '0');
        }else if(isOperator(c) && (operators.CurrentLength()==0 or c=='(')){ //если это оператор и длина 0 или это скобка
            operators.AddEl(c);
        }else{ //если это оператор (не открывающаяся скобка) в непустом стеке операторов
            char operator1,operator2 = c;
            operators.TakeEl(operator1);
            if(((precedence[operator1]>precedence[operator2]) || operator1=='(')&& operator2!=')'){ //если приоритет того что было в стеке выше
                operators.AddEl(operator1);
                operators.AddEl(operator2);
            }else if( c != ')' && precedence[operator1]<=precedence[operator2]){ //если это не закрывающаяся скобка и приоритет того что было ниже или такой же
                //вычислим a oper b
                int b;
                operands.TakeEl(b);
                int a;
                operands.TakeEl(a);

                operands.AddEl(doOper(a,operator1,b));
                operators.AddEl(c);

            }else{ //закрывающаяся скобка выталкивает все до открывающейся скобки
                //в operator 1 лежит верхушка стека, в 2 - ")"
                while(operator1!='('){
                    operator2 = operator1;

                    //действуем
                    int b;
                    operands.TakeEl(b);
                    int a;
                    operands.TakeEl(a);

                    operands.AddEl(doOper(a,operator2,b));


                    operators.TakeEl(operator1);
                }
            }

        }
    }
    while(operators.CurrentLength()>0){
        char oper;
        operators.TakeEl(oper);
        int b;
        operands.TakeEl(b);
        int a;
          operands.TakeEl(a);
          if(oper==')'){
              operators.TakeEl(oper);
              while(oper!='('){
                  operands.AddEl(doOper(a,oper,b));
                  operands.TakeEl(b);
                  operands.TakeEl(a);
                  operators.TakeEl(oper);
                  if(oper=='('){
                      operators.TakeEl(oper);
                      operands.AddEl(doOper(a,oper,b));
                      break;
                  }
              }
```

```
114.             char trash;
115.             operators.TakeEl(trash);
116.
117.         }else{
118.             operands.AddEl(doOper(a,oper,b));
119.         }
120.     }
121.     int result;
122.     operands.TakeEl(result);
123.     return result;
124.
125. }
126.
127. /*
128.  *(int(*)[5])operands.a
129.  * test lane
130.  * 2+3=5 PASS
131.  * 2+1-6/(1+2) = 1 PASS
132.  * 1+1*9+(2^4-6^2) = -10 PASS
133.  * 2^2^2 = 16 PASS
134.  */
135.
136. int main() {
137.     system("chcp 65001"); //переключаем кодировку в кириллицу
138.     //setlocale(LC_ALL, "Russian");
139.     int choice1;
140.
141.     cout<<"Смольников Алексей. Практическая работа 7. Вариант 20"<<endl<<endl;
142.     cout<<"Подсчитать значение инфиксного выражения, реализуя стек на однонаправленном
    списке"<<endl;
143.     cout<<"1 - ввод выражения, 0 - выход"<<endl;
144.
145.     do {
146.         cin >> choice1;
147.
148.         if (choice1 != 1 && choice1 != 0) cout << "Некорректный выбор.\n";
149.     } while (choice1 != 1 && choice1 != 0);
150.
151.     string expression;
152.     switch (choice1) {
153.         case 1:
154.             cout<<"Введите выражение: "<<endl;
155.             cin.ignore(32767, '\n');
156.             getline(cin, expression);
157.             cout<<"Результат: "<<calculateInfixForm(expression);
158.             break;
159.         case 0:
160.             cout<<"Завершение программы...";
161.             break;
162.         default:
163.             break;
164.
165.     }
166.     return 0;
167. }
168.
```

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 (20)