



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования

“МИРЭА - Российский технологический университет”

РТУ МИРЭА

Институт информационных технологий (ИТ)
Кафедра математического обеспечения и стандартизации
информационных технологий (МОСИТ)

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ №8

по дисциплине

«Структуры и алгоритмы обработки данных»

Тема «Стек и очередь»

Выполнил студент группы ИКБО-50-23

Русаков М.Ю.

Принял старший преподаватель

Скворцова Л.А.

Москва 2024

Оглавление

1. Условие задачи	3
2. Постановка задачи	3
3. Задание 1	4
3.1. Подзадание 1	4
3.2. Подзадание 2	6
3.3. Подзадание 3	7
3.4. Подзадание 4	7
4. Задание 2	8
4.1. АТД задачи	8
4.2. Stack	8
4.3. Element	9
4.4. Код реализации АТД	9
4.5. Код основной программы	11
4.6. Тестирование	13
5. Выводы	15

1. Условие задачи

Требуется выполнить два задания.

Первое задание не требует разработки кода, его цель в изучении АТД алгоритмов применения стека и очереди в задачах преобразования выражений из одной формы в другую. Требуется решить три задачи.

Приведены образцы оформления решения каждой задачи. Теоретический материал по алгоритмам преобразования арифметического выражения, представленного в строковом формате, из инфиксной формы в польскую постфиксную или префиксную.

Во втором задании требуется разработать программу вычисления значения арифметического выражения, представленного в одной из трех форм.

2. Постановка задачи

Задания моего персонального варианта (№22):

Задание 1. Выполнить четыре задачи, определенные вариантом задания 1. Варианты задач представлены в таблице 18.

Задание 2. Дано арифметическое выражение в нотации, указанной в варианте, представленное в строковом формате. Разработать программу выполнения задачи варианта. Структура представления стека или очереди определена вариантом.

3. Задание 1

3.1. Подзадание 1

Выполнить преобразование инфиксной записи выражения в постфиксную нотацию, расписывая процесс по шагам.

Исходное выражение (согласно варианту, 22 строка 1 столбца таблицы 18):

$$S = z^{\wedge}(y+x)/m/n^{*}(k-p)$$

Приведем решение задачи в виде последовательности выкладок. Результат выкладок будем записывать в строку S_1 , а операторы – в стек W .

1. Добавим операнд z в строку S_1

S_1 : z

W :

2. Добавим оператор \wedge в стек W

S_1 : z

W : \wedge

3. Добавим открывающую скобку в стек W

S_1 : z

W : $\wedge($

4. Добавим операнд y в строку S_1

S_1 : zy

W : $\wedge($

5. Добавим оператор $+$ в стек W

S_1 : zy

W : $\wedge(+$

6. Добавим операнд x в строку S_1

S_1 : zyx

W : $\wedge(+$

7. Текущий символ равен закрывающей скобке, значит извлекаем все операторы из стека W до открывающей скобки и помещаем в строку

S_1

S_1 : $zyx+$

W : \wedge

8. Извлечем оператор \wedge из стека W и добавим в строку S_1

S_1 : $zyx+\wedge$

W :

9. Добавим оператор $/$ в стек W
 $S_1: зух+\wedge$
 $W: /$
10. Добавим операнд m в строку S_1
 $S_1: зух+\wedge m$
 $W: /$
11. Добавим оператор $/$ в стек W
 $S_1: зух+\wedge m$
 $W: //$
12. Извлечем из стека W оператор $/$ и добавим его в строку S_1
 $S_1: зух+\wedge m/$
 $W: /$
13. Добавим операнд n в строку S_1
 $S_1: зух+\wedge m/n$
 $W: /$
14. Извлечем оператор $/$ из стека W и добавим в строку S_1
 $S_1: зух+\wedge m/n/$
 $W:$
15. Добавим оператор $*$ в стек W
 $S_1: зух+\wedge m/n/$
 $W: *$
16. Добавим открывающую скобку в стек W
 $S_1: зух+\wedge m/n/$
 $W: *($
17. Добавим операнд k в строку S_1
 $S_1: зух+\wedge m/n/k$
 $W: *($
18. Добавим оператор $-$ в стек W
 $S_1: зух+\wedge m/n/k$
 $W: *(-$
19. Добавим операнд r в строку S_1
 $S_1: зух+\wedge m/n/kp$
 $W: *($
20. Текущий символ равен закрывающей скобке, значит извлекаем все операторы из стека W до открывающей скобки и помещаем в строку S_1
 $S_1: зух+\wedge m/n/kp-$
 $W: *$

21. Извлечем из стека W оператор $*$ и добавим его в строку S_1

$S_1: zyx+\wedge m/n/kp-*$

$W:$

Преобразование завершено, поскольку стек W пуст. Полученная строка $S_1 = zyx+\wedge m/n/kp-*$ представляет собой постфиксную форму записи исходного инфиксного выражения.

3.2. Подзадание 2

Представить инфиксную нотацию выражения с расстановкой скобок, расписывая процесс по шагам.

Исходное выражение (согласно варианту, 22 строка 2 столбца таблицы 18):

$$S = afbc^*-zx-/y++$$

Приведем решение задачи в виде таблицы (см. табл. 1). Результат решения запишем в строку S_1 .

Таблица 1 – Решение позадания 2

Стек операндов	Операция (операнд) исходного выражения
a	a
af	f
afb	b
afbc	c
af(b*c)	*
a(f-b*c)	-
a(f-b*c)z	z
a(f-b*c)zx	x
a(f-b*c)(z-x)	-
a(f-b*c)/(z-x)	/
a(f-b*c)/(z-x)y	y
a(((f-b*c)/(z-x))+y)	+
a+(((f-b*c)/(z-x))+y)	+

Преобразование завершено. Полученная строка $S_1 = a+(((f-b*c)/(z-x))+y)$ представляет собой инфиксную форму записи исходного постфиксного выражения.

3.3. Подзадание 3

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

Приведем решение подзадания в виде последовательности выкладок.

1. Преобразуем самое вложенное выражение в префиксную форму: $f-b*c = -f*bc$. Подставим полученное выражение обратно в исходную строку: $a+(((f*bc)/(z-x))+y)$.
2. Аналогично преобразуем $z-x$: $-zx$. Подставим полученное в исходную строку: $a+(((f*bc)/-zx)+y)$.
3. Аналогично преобразуем выражение $(f*bc)/-zx = /-f*bc-zx$. Подставим полученное в исходную строку: $a + ((/-f*bc-zx) + y)$.
4. Аналогично преобразуем выражение $(/-f*bc-zx) + y = +/-f*bc-zxy$. Подставим результат в исходную строку: $a+(+/-f*bc-zxy)$.
5. Аналогично преобразуем выражение: $+a+/-f*bc-zxy$.

Полученное выражение $+a+/-f*bc-zxy$ является префиксной формой записи инфиксного выражения, полученного в подзадании 2.

3.4. Подзадание 4

Вычислить значение выражения.

Исходное выражение (согласно варианту, 22 строка 3 столбца таблицы 18):

$$S = -+3+5,1/*2,4^1+2,6$$

Приведем решение подзадания 4 в виде последовательности выкладок. Начнем с самого внешнего оператора и будем двигаться внутрь

1. Самый внешний оператор — $-$.
2. Разберем выражение $+3+5,1$. Поскольку все операторы являются плюсами, просто найдем сумму всех чисел: $3 + 5 + 1 = 9$.
3. Далее разберем оставшееся выражение $/*2,4^1+2,6$. Самая вложенная операция — $+2,6 = 2 + 6 = 8$. Далее по вложенности следует операция 1,8 (8 — результат сложения на предыдущем шаге). $^1,8 = 1^8 = 1$. Затем следует операция $*2,4 = 2 \cdot 4 = 8$. Последняя операция — частное ($/$) результатов, полученных на последнем и предпоследнем шагах соответственно: $/8,1 = 8 \div 1 = 8$.
4. Остается последняя операция — разность результатов, полученных после выполнения шагов 2 и 3 соответственно: $-9,8 = 9 - 8 = 1$.

В результате вычислений получим ответ: 1.

4. Задание 2

Дано арифметическое выражение в постфиксной нотации, представленное в строковом формате. Разработать программу выполнения задачи варианта. В качестве структуры данных использовать стек, реализованный на основе линейного списка.

4.1. АТД задачи

4.2. Stack

АТД Stack

{

Данные (описание свойств структуры данных задачи)

top – указатель на вершину стека

Операции (объявления операций)

1. Метод, создающий пустой стек

Предусловие: нет

Постусловие: пустой стек

Заголовок: **Stack()**

2. Метод, осуществляющий добавление элемента на вершину стека

Предусловие: value – значение нового элемента

Постусловие: обновленный стек

Заголовок: **void push(int value)**

3. Метод, осуществляющий удаление элемента с вершины стека

Предусловие: исходный стек

Постусловие: обновленный стек

Заголовок: **void pop()**

4. Метод, осуществляющий считывание значения элемента, находящегося на вершине стека

Предусловие: исходный стек

Постусловие: значение элемента, находящегося на вершине стека

Заголовок: **int peek()**

5. Метод, осуществляющий проверку стека на пустоту

Предусловие: исходный стек

Постусловие: true, если стек пуст, иначе false

Заголовок: **bool empty()**

}

4.3. Element

АТД Element

{

Данные (описание свойств структуры данных задачи)

value – значение текущего элемента стека

next – указатель на следующий элемент стека

Операции (объявления операций)

1. Метод, создающий элемент стека

Предусловие: value – значение элемента стека

Постусловие: элемент стека

Заголовок: Element(int value)

}

4.4. Код реализации АТД

1. Код файла stack.h

```
#include "element.h"
```

```
#pragma once
```

```
struct Stack {
```

```
private:
```

```
    // Указатель на вершину стека
```

```
    Element* top;
```

```
public:
```

```
    // Конструктор по умолчанию
```

```
    Stack();
```

```
    // Добавление элемента на вершину стека
```

```
    void push(int value);
```

```
    // Удаление элемента с вершины стека
```

```
    void pop();
```

```
    // Считывание элемент с вершины стека
```

```
    int peek();
```

```

        // Проверка на пустой стек
        bool empty();
};

```

2. Код файла stack.cpp

```

#include "../headers/stack.h"

#include <cstdlib>

Stack::Stack() {
    top = nullptr;
}

void Stack::push(int value) {
    Element* newTop = new Element(value);

    newTop->next = top;
    top          = newTop;
}

void Stack::pop() {
    Element* temp = top->next;

    delete top;

    top = temp;
}

int Stack::peek() {
    if (empty()) {
        // Возврат максимально возможного int (2^31 - 1)
        return INT32_MAX;
    }

    return top->value;
}

bool Stack::empty() {
    if (!top) {
        return true;
    }

    return false;
}

```

3. Код файла element.h

```

#include <string>

#pragma once

struct Element {
    int value;
    Element* next;

    Element(int value);
};

```

4. Код файла element.cpp

```

#include "../headers/element.h"

Element::Element(int value) {
    this->value = value;
    this->next = nullptr;
}

```

4.5. Код основной программы

1. Код файла main.cpp

```

#include "headers/stack.h"
#include "headers/operations.h"

#include <string>
#include <iostream>

int main() {
    std::string OP_STRING;
    std::cin >> OP_STRING;

    int answer = execute(OP_STRING);

    std::cout << "Ответ: " << answer << '\n';
}

```

2. Код файла operations.cpp

```

#include "../headers/operations.h"
#include "../headers/stack.h"

#include <iostream>
#include <string>

bool isOperation(char op) {
    const char patterns[6] = "+-*/^";
}

```

```

    for (int i{}; i < 5; i++) {
        if (patterns[i] == op) {
            return true;
        }
    }

    return false;
}

int operate(int left, int right, char op) {
    switch (op) {
        case '+':
            return left + right;
        case '-':
            return left - right;
        case '*':
            return left * right;
        case '/':
            return left / right;
        default:
            int result{1};

            for (int i{}; i < right; i++) {
                result *= left;
            }

            return result;
    }
}

int execute(std::string OP_STRING) {
    const int ASCII_SHIFT = 48;

    Stack elems;
    for (char& sym: OP_STRING) {
        if (isOperation(sym)) {
            int right = elems.peek();
            elems.pop();

            int left = elems.peek();
            elems.pop();

            int result = operate(left, right, sym);
            elems.push(result);
        }
        else {

```

```

        if (sym == ',') { continue; }
        else {
            // Приведение char к int путем сдвига ASCII кода символа
на 48            elems.push(int(sym) - ASCII_SHIFT);
        }
    }
}

return elems.peek();
}

```

4.6. Тестирование

Тест №1. Входная строка: $8,2,5*+4,3,2*-4- /$. Ожидаемые выходные данные: “Ответ: -3 ”



Рисунок 1 – Результаты теста №1

Тест №2. Входная строка: $4,8,2^*-5+$. Ожидаемые выходные данные: “Ответ: -7 ”



Рисунок 2 – Результаты теста №1

Тест №3. Входная строка: $2,2,1+^4/2/9,6-^*$. Ожидаемые выходные данные: “Ответ: 3 ”



Рисунок 3 – Результаты теста №3

5. Выводы

В результате выполнения работы были получены знания и навыки по реализации структуры стек и очередь, выполнению операций управления стеком и очередью, практическому применению стека и очереди при вычислении значения арифметического выражения и преобразования арифметических выражений из инфиксной нотации в польскую нотацию.