

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. Структуры Стек и Очередь

|  |  |  |
| --- | --- | --- |
| Выполнил студент группы ИКБО-65-23 |  | Петров А. |
| Принял старший преподаватель |  | Скворцова Л.А. |

Москва 2023

Оглавление

[2. Задание 1 3](#_Toc167358007)

[2.1. Требование к выполнению задания 1 3](#_Toc167358008)

[2.2. Задача 1 3](#_Toc167358009)

[2.3 Задача 2 6](#_Toc167358010)

[2.4 Задача 3 7](#_Toc167358011)

[2.5 Задание 4 9](#_Toc167358012)

[Задание 2 11](#_Toc167358013)

[3.1 Требование к выполнению задания 2 11](#_Toc167358014)

[3.2 Реализация АТД 11](#_Toc167358015)

[3.3 Код для задания 2 13](#_Toc167358016)

[3.4 Тестирование алгоритма программы 15](#_Toc167358017)

[4. Вывод 16](#_Toc167358018)

[5. Информационные источники 17](#_Toc167358019)

**1. Требование к выполнению практической работы**

Требуется выполнить два задания. В первом задании не требует разработки кода, его цель в изучении 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 и префиксные, вычисление значения выражения, представленного в одной из трех форм представлен в приложении 2 практикума {1]. Во втором задании требуется разработать программу вычисления значения арифметического выражения, представленного в одной из трех форм.

# 2. Задание 1

Выполнить три задачи, определенные вариантом задания 1. Процесс выполнения представить в отчете по зданию 1. Варианты задач представлены в табл. 18.

## 2.1. Требование к выполнению задания 1

1. Изучить: − алгоритмы преобразования арифметического выражения, представленного в инфиксной форме в, обратную польскую запись (постфиксную форму) и префиксную формы и обратно; − алгоритмы вычисления значения арифметического выражения представленного в инфиксной, префиксной, постфиксной формах записи. Алгоритмы преобразований приведены в [11].

2. Выполнить задачу 1.

3. Выполнить задачу 2.

4. Выполнить задачу 3.

5. Выполнить задачу 4.

## 2.2. Задача 1

Выполнить преобразование инфиксной записи выражения S= z^(y+x)/m/n\*(k-p) в постфиксную нотацию, расписывая процесс по шагам.

A+B\*C+D

|  |
| --- |
| Формирование постфиксной записи инфиксного выражения, в котором операции различного приоритета и в подвыражениях одного приоритета. Исходное выражение строка S: z^(y+x)/m/n\*(k-p)  Результат в строке S  w – стек операция  *Решение*  1. Читаем символ z, это операнд, добавляем к выходной строке.  S: z  2. Читаем символ ^, это оператор, помещаем его в стек.  w: ^  S: z  3. Читаем символ (, это открывающая скобка, помещаем её в стек..  w: ^(  S: z  4. Читаем символ y, это операнд, добавляем к выходной строке..  w: ^(  S: zy  5. Читаем символ +, это оператор, помещаем его в стек.  w: ^(+  S: zy  6. Читаем символ x, это операнд, добавляем к выходной строке..  w: ^(+  S: zyx  7. Читаем символ ), это закрывающая скобка, выталкиваем операторы из стека в выходную строку до тех пор, пока не найдём открывающую скобку.  w: ^  S: zyx+  8. Читаем символ /, это оператор, помещаем его в стек (он имеет меньший приоритет, чем ^, так что выталкивать не нужно).  w: ^/  S: zyx+  9. Читаем символ m, это операнд, добавляем к выходной строке.  w: ^/  S: zyx+m  10. Читаем символ /, это оператор, выталкиваем предыдущий оператор / из стека в выходную строку и помещаем новый / в стек.  w: ^/  S: zyx+m/  11. Читаем символ n, это операнд, добавляем к выходной строке.  w: ^/  S: zyx+m/n  12. Читаем символ \*, это оператор, выталкиваем предыдущий оператор / из стека в выходную строку и помещаем новый \* в стек.  w: ^\*  S: zyx+m/n/  13. Читаем символ (, это открывающая скобка, помещаем её в стек.  w: ^\*(  S: zyx+m/n/  14. Читаем символ k, это операнд, добавляем к выходной строке.  w: ^\*(  S: zyx+m/n/k  15. Читаем символ -, это оператор, помещаем его в стек  w: ^\*(-  S: zyx+m/n/k  16. Читаем символ p, это операнд, добавляем к выходной строке.  w: ^\*(-  S: zyx+m/n/kp  17. Читаем символ ), это закрывающая скобка, выталкиваем операторы из стека в выходную строку до тех пор, пока не найдём открывающую скобку.  w: ^\*  S: zyx+m/n/kp-  18. Входная строка закончилась, выталкиваем оставшиеся операторы из стека в выходную строку.  w:  S: zyx+m/n/kp-^\*  **Итог:**  zyx+m/n/kp-^\* |

## 2.3 Задача 2

Представить инфиксную нотацию a f b c\*-z x-/y++ (идентификаторы одно символьные) с расстановкой скобок, расписывая процесс по шагам.

|  |  |
| --- | --- |
| **Стек операндов** | **Операция из исходного выражения** |
| f | 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)) | + |

## 2.4 Задача 3

Представить префиксную нотацию выражения, полученного в результате выполнения задачи 2 (a + (((f - (b \* c)) / (z - x)) + y)), расписывая процесс по шагам.

|  |
| --- |
| Исходное выражение строка S: (a + (((f - (b \* c)) / (z - x)) + y))  Результат в строке S  w – стек операция  *Решение*  Обратим входное выражение: ((y + (x - z) / (c \* b) - f) + a)  1. Читаем (, помещаем в стек.  S:  w: (  2. Читаем (, помещаем в стек.  S:  w: ((  3. Читаем y, добавляем к выходной строке.  S: y  w: ((  4. Читаем +, помещаем в стек.  S: y  w: ((+  5. Читаем (, помещаем в стек.  S: y  w: ((+(  6. Читаем x, добавляем к выходной строке.  S: yx  w: ((+(  7. Читаем -, помещаем в стек.  S: yx  w: ((+(-  8. Читаем символ z, это операнд, добавляем к выходной строке.  S: yxz  w: ((+(-  9. Читаем ), выталкиваем из стека до (  S: yxz-  w: ((+  10. Читаем /, помещаем в стек.  S: yxz-  w: ((+/  11. Читаем (, помещаем в стек.  S: yxz-  w: ((+/(  12. Читаем c, добавляем к выходной строке.  S: yxz-c  w: ((+/(  13. Читаем \*, помещаем в стек.  S: yxz-c  w: ((+/(\*  14. Читаем b, добавляем к выходной строке.  S: yxz-cb  w: ((+/(\*  15. Читаем ), выталкиваем из стека до (.  S: yxz-cb\*  w: ((+/  16. Читаем -, выталкиваем / из стека в выходную строку и помещаем -.  S: yxz-cb\*/  w: ((-  17. Читаем f, добавляем к выходной строке.  S: yxz-cb\*/f  w: ((-  18. Читаем ), выталкиваем из стека до (.  S: yxz-cb\*/f-  w: ((+  19. Читаем +, помещаем в стек.  S: yxz-cb\*/f-  w: ((++  20. Читаем a, добавляем к выходной строке.  S: yxz-cb\*/f-a  w: ((++  21. Входная строка закончилась, выталкиваем оставшиеся операторы из стека.  S: yxz-cb\*/f-a++  w:  Обратим полученное постфиксное выражение: ++a-f\*/bcz-xy.  **Итог:**  ++a-f\*/bcz-xy |

## 2.5 Задание 4

Вычислить значение выражения -+3+5 1/\*2 4^1+ 2 6

В данном случае выражение представляется в префиксной форме, так как операторы расположены перед операндами.

|  |  |  |
| --- | --- | --- |
| Элемент выражения | Состояние строки | Состояние стека  (вершина справа) |
| 6 | -+3+5 1/\*2 4^1+2 | 6 |
| 2 | -+3+5 1/\*2 4^1+ | 2,6 |
| + | -+3+5 1/\*2 4^1 | (2 + 6) = 8 |
| 1 | -+3+5 1/\*2 4^ | 1, 8 |
| ^ | -+3+5 1/\*2 4 | (1 ^ 8) = 1 |
| 4 | -+3+5 1/\*2 | 4, 1 |
| 2 | -+3+5 1/\* | 2, 4, 1 |
| \* | -+3+5 1/ | (2 \* 4) = 8, 1 |
| / | -+3+5 1 | (8 / 1) = 8 |
| 1 | -+3+5 | 1, 8 |
| 5 | -+3+ | 5, 1, 8 |
| + | -+3 | (5 + 1) = 6, 8 |
| 3 | -+ | 3, 6, 8 |
| + | - | (3 + 6) = 9, 8 |
| - |  | (9 - 8) = 1 |
|  |  | 1 |

Ответ: 1

# Задание 2

## 3.1 Требование к выполнению задания 2

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 (или методы класса).
4. Выполнить тестирование разработанного приложения.
5. Оформить отчет.

|  |  |  |  |
| --- | --- | --- | --- |
| № | Нотация  представления выражения | Структура  реализации стека или очереди | Задача |
| 22 | Постфиксная | Список | Вычислить значение выражения |

## 3.2 Реализация АТД

|  |
| --- |
| АТД Stack{ Данные:  struct Node - Структура узла списка  Методы:  isEmpty() – проверка стека на пустоту  add () – вставка элемента в стек  del () – извлечение элемента из стека  peek () – получение верхнего элемента из стека  evaluatePostfix() - вычисления значения выражения в постфиксной нотации |

Операции:

|  |
| --- |
| // Операция, отвечающая за проверку стека на пустоту  //постусловие: возвращено булевое значение в зависимости от того пустой ли стек  isEmpty (); |

|  |
| --- |
| // Операция, отвечающая за вставку элемента в стек  //предусловие: val – входные данные  //постусловие: возвращено булевое значение в зависимости от того пустой ли стек  add(value); |

|  |
| --- |
| // Операция, отвечающая за извлечение элемента из стека  //постусловие: последний элемент из стека удален  del(); |

|  |
| --- |
| // Операция, отвечающая за получение верхнего элемента стека  //постусловие: возвращен верхний элемент стека, если он есть peek(value); |

|  |
| --- |
| // Функция для вычисления значения выражения в постфиксной нотации  //предусловие: expression– входные данные  //постусловие: возвращает значение выражения  evaluatePostfix(expression); |

## 3.3 Код для задания 2

|  |
| --- |
| #include <iostream>  #include <vector>  #include <string>  using namespace std;  // Структура узла списка  struct Node {  int data;  Node\* next;  Node(int val) : data(val), next(nullptr) {}  };  // Класс стека на основе списка  class Stack {  private:  Node\* top;  public:  Stack() : top(nullptr) {}  // Проверка на пустоту  bool isEmpty() {  return top == nullptr;  }  // Добавление элемента в стек  void add(int val) {  Node\* newNode = new Node(val);  newNode->next = top;  top = newNode;  }  // Удаление элемента из стека  void del() {  if (isEmpty()) {  cout << "Stack is empty. Cannot delete." << endl;  return;  }  Node\* temp = top;  top = top->next;  delete temp;  }  // Получение верхнего элемента стека без удаления  int peek() {  if (isEmpty()) {  cout << "Stack is empty. Cannot peek." << endl;  return -1;  }  return top->data;  }  };  // Функция для вычисления значения выражения в постфиксной нотации  int evaluatePostfix(string expression) {  Stack stack;  for (char& c : expression) {  if (isdigit(c)) {  stack.add(c - '0');  }  else {  int operand2 = stack.peek();  stack.del();  int operand1 = stack.peek();  stack.del();  switch (c) {  case '+':  stack.add(operand1 + operand2);  break;  case '-':  stack.add(operand1 - operand2);  break;  case '\*':  stack.add(operand1 \* operand2);  break;  case '/':  stack.add(operand1 / operand2);  break;  }  }  }  return stack.peek();  }  int main() {  // Запуск тестирования  string exp = "34+2/1";  cout << "Expression: " << exp << endl;  cout << "Result: " << evaluatePostfix(exp) << endl;  return 0;  } |

## 3.4 Тестирование алгоритма программы

|  |  |  |
| --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат |
| 1 | 34+2/1 | 1 |
| 2 | 3333\*\*\* | 81 |



Результат первого тестирования



Результат второго тестирования

# 4. Вывод

В ходе выполнения задания 1 для 5 практической работы я освоил библиотеку string.h, работу с свободными массивами \*\*char, а также преобразования строки в нуль строку. А уже во время выполнения задания 2 я освоил современные способы работы со строками, добавленные в с++, и можно сделать вывод, что используя современные способы код может быть в разы читабельнее и меньше.

# 5. Информационные источники

1. Приложение к практическим работам – СДО (online-edu.mirea.ru)

2. Структуры и алгоритмы обработки данных – методические указания / Скворцова Л.А., Филатов А.С., Гусев К.В., Трушин С.М. - Москва, МИРЭА - Российский технологический университет, 2023.