

|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования  **"МИРЭА - Российский технологический университет"**  **РТУ МИРЭА** |

|  |  |
| --- | --- |
| Отчет по выполнению практического задания №8 | |
| **Тема**: Нотации выражений | |
| **Дисциплина:** Структуры и алгоритмы обработки данных | |
| Выполнил студент | Антонов А.Д. |
| Группа | ИКБО-01-20 |

Москва 2021

# Оглавление

1. [Задание 1 (Вариант №1) 3](#_bookmark0)
   1. [Процесс выполнения упражнения 1 3](#_bookmark1)
   2. [Процесс выполнения упражнения 2 4](#_bookmark2)
   3. [Процесс выполнения упражнения 3 5](#_bookmark3)
   4. [Процесс выполнения упражнения 4 8](#_bookmark4)
2. [Программная реализация задания 2 8](#_bookmark5)
   1. [Разработать функцию(функции) преобразования инфиксной формы](#_bookmark6) [выражения в постфиксную форму 8](#_bookmark6)
      1. [Постановка задачи 8](#_bookmark7)
      2. [Код программы 8](#_bookmark8)
      3. [Тестирование: 12](#_bookmark9)
   2. [Реализовать операции стек: втолкнуть элемент в стек, вытолкнуть](#_bookmark10) [элемент из стека, вернуть значение элемента в вершине стека, сделать стек](#_bookmark10) [пустым, определить пуст ли стек. 12](#_bookmark10)
      1. [Постановка задачи 12](#_bookmark11)
      2. [Код программы 12](#_bookmark12)
   3. [Реализация структуры из двух стеков внутри одного массива 14](#_bookmark13)
      1. [Постановка задачи 15](#_bookmark14)
      2. [Код функции и программы 15](#_bookmark15)

[Выводы 17](#_bookmark16)

[Список информационных источников 17](#_bookmark17)

# Задание 1 (Вариант №1)

## Процесс выполнения упражнения 1

Условие задания:

Провести преобразование инфиксной записи выражения в префиксную нотацию, расписывая процесс по шагам для выражения: *S=a+(b-c\*k)/(d\*e-f)*

При переводе в префиксную запись выражение читается справа налево, а формируется в обратном порядке, при этом каждый новый символ добавляется в начало.

Таблица 1. Алгоритм преобразования выражения S

|  |  |  |  |
| --- | --- | --- | --- |
| Символ | Стек | Вывод | Комментарий |
| ) | ) |  | Оператор добавляется в стек |
| f | ) | f | Операнд добавляется в начало |
| - | -) | f | Оператор с наивысшим приоритетом  добавляется в стек |
| e | -) | ef | Операнд добавляется в начало |
| \* | -\*) | ef | Оператор с наивысшим приоритетом  добавляется в стек |
| d | -\*) | def | Операнд добавляется в начало |
| ( |  | -\*def | Операторы выталкиваются до  открывающей скобки |
| / | / | -\*def | Оператор добавляется в стек |
| ) | )/ | -\*def | Оператор добавляется в стек |
| k | )/ | k-\*def | Операнд добавляется в начало |
| \* | \*)/ | k-\*def | Оператор с наивысшим приоритетом  добавляется в стек |
| c | \*)/ | ck-\*def | Операнд добавляется в начало |
| - | -)/ | \*ck-\*def | - имеет меньший приоритет, поэтому  выталкивается \*, после добавляется - |
| b | -)/ | b\*ck-\*def | Операнд добавляется в начало |
| ( | / | -b\*ck-\*def | Операторы выталкиваются внутри  скобок |
| + | + | /-b\*ck-\*def | + имеет меньший приоритет, поэтому  выталкивается /, после добавляется - |

|  |  |  |  |
| --- | --- | --- | --- |
| a | + | a/-b\*ck-\*def | Операнд добавляется в начало |

В конце добавляем из стека оставшиеся операторы:

*S =+ a / - b \* c k - \* d e f*

## Процесс выполнения упражнения 2

Условие задания:

Представить инфиксную нотацию выражений (идентификаторы односимвольные)

* x y z \* a b d / - + c - +
* x y z \* d / + a b \* c o k - / - -

Таблица 2. Алгоритм преобразования выражения 1

|  |  |
| --- | --- |
| Выражение | Комментарий |
| x y z \* a b d / - + c - + | Идём по выражению слева направо до первого оператора — \*, далее применяем его к двум операндам,  стоящим справа от него |
| x y\*z a b d / - + c - + | Аналогично первому шагу находим операнд / применяем его аналогично. Действуем так, пока все операторы не  встанут на свои места |
| x y\*z a b/d - + c - + | Прим*. операторы и операнды,которые поставлены на свои места*  *записываются без пробела* |
| x y\*z a-b/d + c - + |  |
| x y\*z+a-b/d c - + |  |
| x y\*z+a-b/d-c + |  |
| x+y\*z+a-b/d-c |  |

Инфиксная запись выражения 1: *S1 = x + y\*z + a - b/d - c*

Аналогично первому выражению преобразуем второе

Таблица 2. Алгоритм преобразования выражения 2

|  |  |
| --- | --- |
| Выражение | Комментарий |
| x y z \* d / + a b \* c o k - / - - |  |
| x y\*z d / + a b \* c o k - / - - |  |
| x y\*z/d + a b \* c o k - / - - |  |
| x+y\*z/d a b \* c o k - / - - |  |
| x+y\*z/d a\*b c o k - / - - |  |
| x+y\*z/d a\*b c o-k / - - |  |
| x+y\*z/d a\*b c o-k / - - |  |
| x+y\*z/d a\*b c/(o-k) - - |  |
| x+y\*z/d a\*b-c/(o-k) - |  |
| x+y\*z/d-a\*b-c/(o-k) |  |

Инфиксная запись выражения 2: *S2 = x + y \* z / d – a \* b – c / (o – k)*

## Процесс выполнения упражнения 3

Условие задания:

Представить префиксную нотацию выражений, указанных в пункте 1.2.

Таблица 4. Алгоритм преобразования выражения 1

|  |  |  |
| --- | --- | --- |
| Символ | Стек | Вывод |
| c |  | c |
| - | - |  |
| d | - | dc |
| / | /- | dc |
| b | /- | bdc |
| - | - | -/bdc |
| a | - | a-/bdc |
| + | + | -a-/bdc |
| z | + | z-a-/bdc |
| \* | \*+ | z-a-/bdc |
| y | \*+ | yz-a-/bdc |
| + | + | \*+ yz-a-/bdc |
| x | + | x\*+ yz-a-/bdc |
|  |  | + x\*+ yz-a-/bdc |

Таблица 4. Алгоритм преобразования выражения 2

|  |  |  |
| --- | --- | --- |
| Символ | Стек | Вывод |
| ) | ) |  |
| k | ) | k |
| - | -) | k |
| o | -) | ok |
| ( |  | -ok |
| / | / | -ok |
| c | / | c-ok |
| - | - | /c-ok |
| b | - | b/c-ok |
| \* | \*- | b/c-ok |
| a | \*- | ab/c-ok |
| - | - | \*- ab/c-ok |
| d | - | d\*- ab/c-ok |
| / | /- | d\*- ab/c-ok |
| z | /- | zd\*- ab/c-ok |
| \* | \* | /-zd\*- ab/c-ok |
| y | \* | y/- zd\*- ab/c-ok |
| + | + | \*y/-zd\*-ab/c-ok |
| x | + | x\*y/-zd\*-ab/c-ok |
|  |  | + x\*y/-zd\*-ab/c-ok |

## Процесс выполнения упражнения 4

Условие задания:

Провести вычисление значения выражения, представленного в постфиксной форме, расписывая процесс по шагам: 7 2 3\*5 8 2/-+1-+

Таблица 6. Процесс вычислений

|  |  |
| --- | --- |
| Выражение | Вычисление |
| 7 2 3\*5 8 2 / - + 1 - + |  |
| 7 2\*3 5 8 2 / - + 1 - + | 2\*3=6 |
| 7 6 5 8/2 - + 1 - + | 8/2=4 |
| 7 6 5-4 + 1 - + | 5-4=1 |
| 7 6+1 1 - + | 6+1=7 |
| 7 7-1 + | 7-1=6 |
| 7+6 | 7+6=13 |
| 13 |  |

Значение выражения = 13.

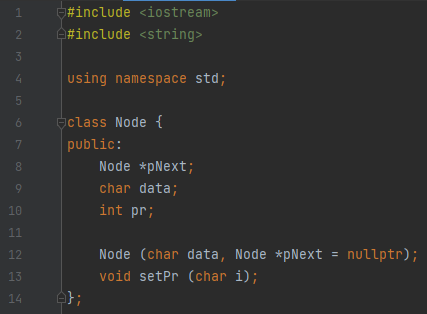
# Программная реализация задания 2

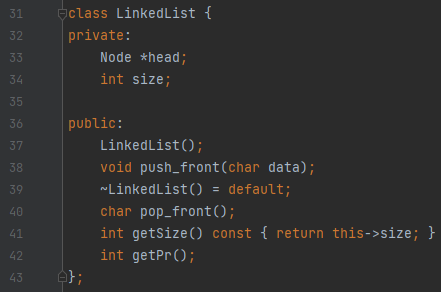
## Разработать функцию (функции) преобразования инфиксной формы выражения в постфиксную форму.

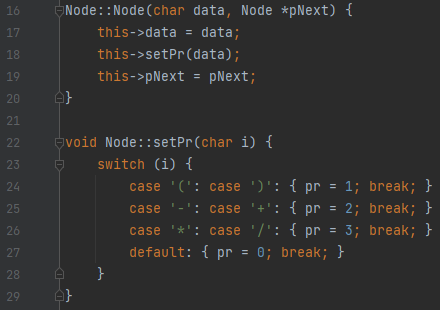
## Постановка задачи:

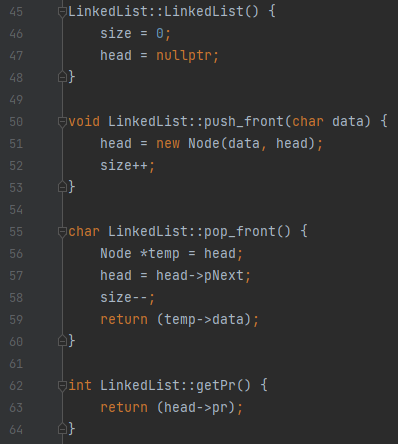
Выбрать структуру данных для реализации стека и описать функции работы со стеком, реализовать функцию, преобразования инфиксной формы выражения в постфиксную форму.

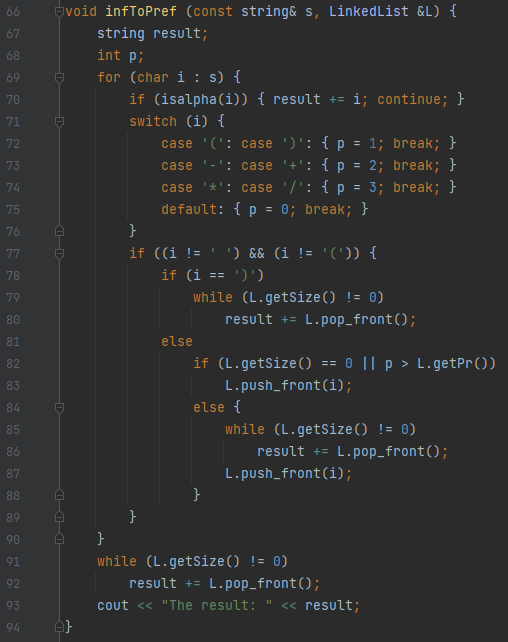
## Код программы:

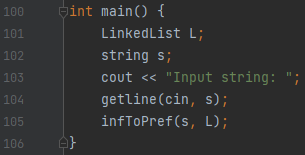












## Тестирование:

|  |  |  |  |
| --- | --- | --- | --- |
| Номер теста | Входные данные | Ожидаемый результат | Результат программы |
| 1 | x+y\*z+a-b/d-c | xyz\*+a+bd/-c- |  |
| 2 | x+y\*z/d-a\*b-c/o | xyz\*+d/ab\*-co/- |  |
| 3 | a+b/c+d+e-f/m\*k | abc/+d+e+fm/-k\* |  |

## Реализовать операции стек: втолкнуть элемент в стек, вытолкнуть элемент из стека, вернуть значение элемента в вершине стека, сделать стек пустым, определить пуст ли стек.

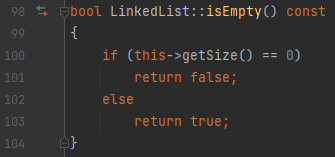
## Постановка задачи:

* Создать класс или просто заголовочный файл с функциями.
* Применить операции для вычисления значения выражения п.4 данного варианта.

Так как стек реализован с помощью односвязного списка, код для него можно взять из предыдущих практик. Единственный новый метод – определение пуст ли стек.

## Код программы

Код для нового метода:



Алгоритм main() для проверки работы функции isEmpty:

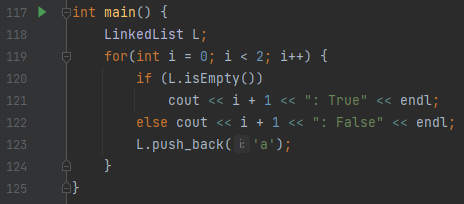
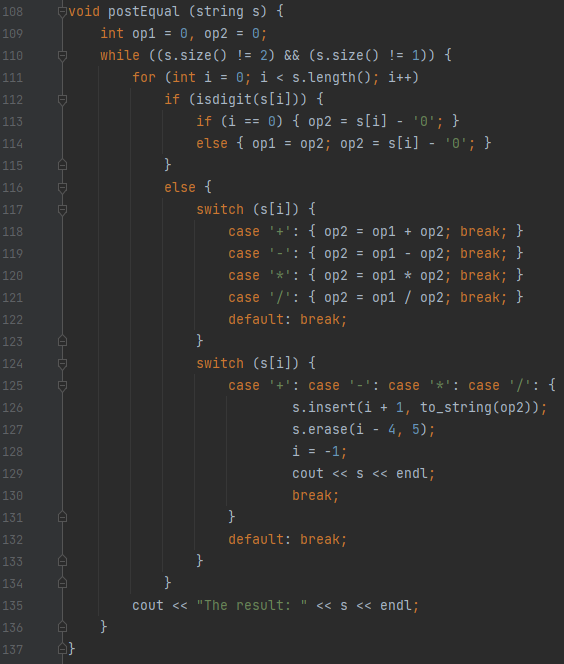




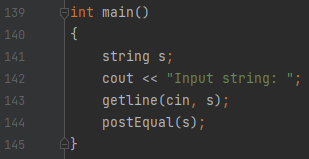
Рис.1 Результат работы программы

Из рис. 1 мы видим, что функция isEmpty работает корректно.

Для реализации вычисления значения выражения п.4 данного варианта напишем функцию, которая будет вычислять значение выражения в постфиксной записи:



Алгоритм функции main():



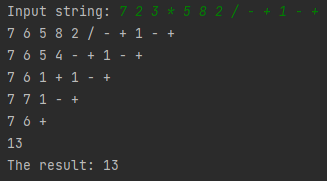


Рис.2 Результат работы программы

Результат программы совпадает с вычислениями, сделанными нами в пункте 1.4, следовательно, программа работает корректно.

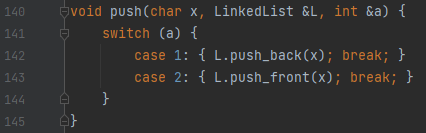
# Реализация структуры из двух стеков внутри одного массива

## Постановка задачи:

Реализовать стек на следующей структуре: хранить два стека в одном массиве, когда один располагается в начале массива и растет к концу массива, а второй располагается в конце и растет к началу. Реализуйте операцию Push(x,S) –втолкнуть элемент х в стек S, где S один или другой стек.

Так как стек реализован с помощью односвязного списка, код для него можно взять из предыдущих практик. Процесс добавления и вывода всех элементов реализуем через меню.

## Код функции и программы:



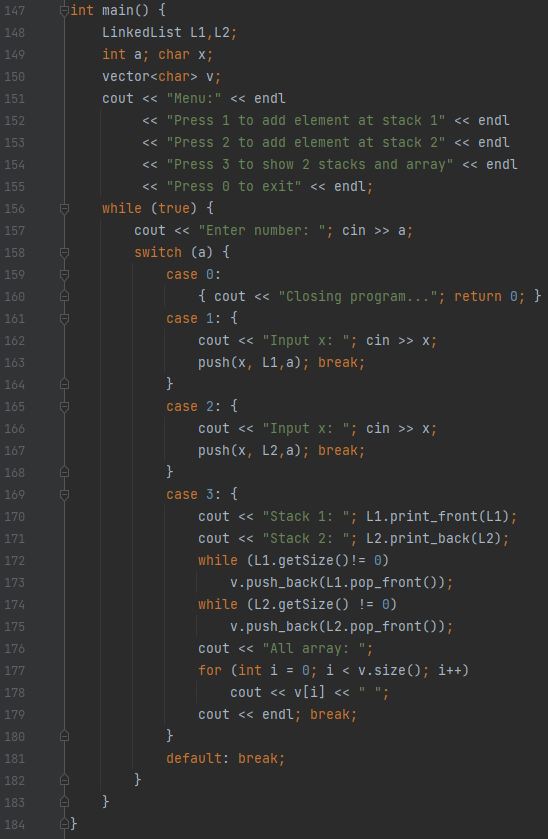




Рис.3 Результат работы программы

Из рис.3 мы видим, что программа работает корректно.

# Выводы

В ходе данной практической работы была разработана ещё одна динамическая структура данных – стек. За основу была взята другая динамическая структура данных, рассмотренная в одной из прошлых практических работ – односвязный список. Были разработаны методы для работы со стеком: метод для вставки нового элемента в стек, метод для извлечения элемента из стека, метод для проверки стека на пустоту, метод (деструктор) для очисти стека и освобождения выделенной памяти. Были изучены постфиксная и префиксная нотации, а также вычислены значения выражений, записанных в виде этих нотаций. Была разработана функция для сложения больших чисел, выходящих за рамки стандартных форматов. Все разработанный функции и методы прошли проверку на корректность результата.

# Список информационных источников

1. Лекционный материал по структурам и алгоритмам обработки данных Алпатова А.Н.
2. Math.semestr – Обратная польская запись - https://www.wikidocs.ru/preview/59279/11 (дата обращения: 29.05.2021)
3. YouTube – Реализация односвязного списка C++ на канале #SimpleCode (Уроки №133-135) - https://www.youtube.com/watch?v=SajrPhE6FoQ (дата обращения: 25.04.2021).