Министерство цифрового развития, связи и массовых коммуникаций

Ордена Трудового Красного Знамени федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Отчёт по лабораторной работе № 4

«Реализация стека/дека»

по дисциплине «Системы и алгоритмы обработки данных»

Выполнил: студент группы БВТ1905

Ахрамешин Алексей Сергеевич

Проверил:

Павликов Артем Евгеньевич

Москва

2021

Оглавление

[Цель работы 3](#_Toc72360231)

[Выполнение 5](#_Toc72360232)

[Снимки экрана выполнения программы 12](#_Toc72360233)

[Вывод 14](#_Toc72360234)

# Цель работы

В ходе выполнения лабораторной работы №4 мне необходимо изучить и реализовать стек и дек, впоследствии реализовать алгоритмы решения задач с их использованием

●  **Стек** (stack):  
операции *для стека*: инициализация, проверка на пустоту, добавление нового элемента в начало, извлечение элемента из начала;

●  **Дек** (двусторонняя очередь, deque):  
операции *для дека*: инициализация, проверка на пустоту, добавление нового элемента в начало, добавление нового элемента в конец, извлечение элемента из начала, извлечение элемента из конца.

Разработать программу обработки данных, содержащихся в заранее подготовленном txt-файле, в соответствии с заданиями, применив указанную в задании структуру данных. Результат работы программы вывести на экран и сохранить в отдельном txt-файле.

Оформить отчет о лабораторной работе в ipynb или pdf-файле. **Задания:**

* 1. Отсортировать строки файла, содержащие названия книг, в алфавитном порядке с использованием двух ***деков*.**
  2. ***Дек*** содержит последовательность символов для шифровки сообщений. Дан текстовый файл, содержащий зашифрованное сообщение. Пользуясь ***деком****,* расшифровать текст. Известно, что при шифровке каждый символ сообщения заменялся следующим за ним в ***деке*** по часовой стрелке через один.
  3. Даны три стержня и *n* дисков различного размера. Диски можно надевать на стержни, образуя из них башни. Перенести *n* дисков со стержня *А* на стержень *С***,** сохранив их первоначальный порядок. При переносе дисков необходимо соблюдать следующие правила:

- на каждом шаге со стержня на стержень переносить только один диск;  
- диск нельзя помещать на диск меньшего размера;  
- для промежуточного хранения можно использовать стержень *В***.**Реализовать алгоритм, используя три ***стека*** вместо стержней *А***,** *В***,** *С***.** Информация о дисках хранится в исходном файле.

* 1. Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс круглых скобок в тексте, используя ***стек*.**
  2. Дан текстовый файл с программой на алгоритмическом языке. За один просмотр файла проверить баланс квадратных скобок в тексте, используя ***дек*.**

1. Дан файл из символов. Используя ***стек*,** за один просмотр файла напечатать сначала все цифры, затем все буквы, и, наконец, все остальные символы, сохраняя исходный порядок в каждой группе символов.
2. Дан файл из целых чисел. Используя ***дек*,** за один просмотр файла напечатать сначала все отрицательные числа, затем все положительные числа, сохраняя исходный порядок в каждой группе.
3. Дан текстовый файл. Используя ***стек*,** сформировать новый текстовый файл, содержащий строки исходного файла, записанные в обратном порядке: первая строка становится последней, вторая – предпоследней и т.д.
4. Дан текстовый файл. Используя ***стек*,** вычислить значение логического выражения, записанного в текстовом файле в следующей форме:  
   < ЛВ > ::= **T** | **F** | (**N**<ЛВ>) | (<ЛВ>**A**<ЛВ>) | (<ЛВ>**X**<ЛВ>) | (<ЛВ>**O**<ЛВ>),  
   где буквами обозначены логические константы и операции:

**T** – True, **F** – False, **N** – Not, **A** – And, **X** – Xor, **O** – Or.

1. Дан текстовый файл. В текстовом файле записана формула следующего вида: <Формула> ::= <Цифра> | **M**(<Формула>,<Формула>) | **N**(Формула>,<Формула>) < Цифра > ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9  
   где буквами обозначены функции:  
   **M** – определение максимума, **N** – определение минимума.  
   Используя ***стек***, вычислить значение заданного выражения.

11.Дан текстовый файл. Используя ***стек***, проверить, является ли содержимое текстового файла правильной записью формулы вида:  
< Формула > ::= < Терм > | < Терм > + < Формула > | < Терм > - < Формула >  
< Терм > ::= < Имя > | (< Формула >)

< Имя > ::= **x** | **y** | **z**

# Выполнение

Код программы:

Задание 1:

const DeQueue = require('./queue');

function task1(array){

let dequeueInc = new DeQueue.DeQueue();

let dequeueSort = new DeQueue.DeQueue();

let sortArr = [];

array.map(item => dequeueInc.pushFront(item))

while(!dequeueInc.isEmpty()){

console.log(dequeueInc.peekFront())

if(dequeueInc.peekFront() <= dequeueSort.peekFront() || dequeueSort.isEmpty()){

dequeueSort.pushFront(dequeueInc.popFront());

}

else{

dequeueInc.pushBack(dequeueSort.popFront());

}

}

while(!dequeueSort.isEmpty()){

sortArr = [...sortArr,dequeueSort.popFront()];

}

return sortArr;

}

Задание 2:

const DeQueue =require('./queue');

let dec = new DeQueue.DeQueue()

let str = `abcdef`;

function cip (str,dec){

let str1 = "";

for(let i=0; i<str.length; i++){

while (str1.length<i+1){

if(dec.peekBack() == str[i]){

dec.pushFront(dec.popBack())

dec.pushFront(dec.popBack())

str1+=dec.peekBack();

}

dec.pushFront(dec.popBack())

}

}

return str1

}

console.log(cip(str,dec))

Задание 3:

const Stack = require(`./stack`)

function transferDisk(a, b){

if (b.isEmpty() === true) {

b.push(a.peek());

a.pop();

return 1;

} else if (a.isEmpty() === true) {

a.push(b.peek());

b.pop();

return 2;

} else {

if (b.peek() > a.peek()) {

b.push(a.peek());

a.pop();

return 1;

} else {

a.push(b.peek());

b.pop();

return 2;

}

}

}

function han (kol){

let s = new Stack.Stack()

let a = new Stack.Stack()

let d = new Stack.Stack()

let n = kol

for (let i = n; i >= 1; i--) {

s.push(i);

}

let x = Math.pow(2, n) - 1

let i = 1

if (n % 2 === 0) {

while (i <= x) {

if (i % 3 === 1) {

let y = transferDisk(s, a)

if (y === 1) {

console.log("Переместить диск " + a.peek() + " с StackA на StackB")

} else

console.log("Переместить диск " + s.peek() + " с StackB на StackA")

} else if (i % 3 === 2) {

let y = transferDisk(s, d)

if (y === 1) {

console.log("Переместить диск " + d.peek() + " с StackA на StackC")

} else

console.log("Переместить диск " + s.peek() + " с StackC на StackA")

} else {

let y = transferDisk(a, d)

if (y === 1) {

console.log("Переместить диск " + d.peek() + " с StackB на StackC")

} else

console.log("Переместить диск " + a.peek() + " с StackC на StackB")

}

i++

}

} else {

while (i <= x) {

if (i % 3 === 1) {

let y = transferDisk(s, d);

if (y === 1) {

console.log("Переместить диск " + d.peek() + " с StackA на StackC")

} else

console.log("Переместить диск " + s.peek() + " с StackC на StackA")

} else if (i % 3 === 2) {

let y = transferDisk(s, a);

if (y === 1) {

console.log("Переместить диск " + a.peek() + " с StackA на StackB")

} else

console.log("Переместить диск " + s.peek() + " с StackB на StackA")

} else {

let y = transferDisk(a, d);

if (y === 1) {

console.log("Переместить диск " + d.peek() + " с StackB на StackC")

} else

console.log("Переместить диск " + a.peek() + " с StackC на StackB")

}

i++;

}

}

return 0;

}

console.log(han(3))

Задание 4:

const Stack = require(`./stack`)

function bracketFinderStack (array){

let stack = new Stack.Stack()

let flag = true;

array.map( item =>{

if(item === '('){

stack.push('(')

}

else if(item === ')'){

if(!stack.isEmpty()){

stack.pop()

}

else {

flag = false

}

}

})

return flag && !!stack.isEmpty()

}

Задание 5:

const DeQueue = require('./queue')

function bracketFinderDeque(array){

let deque = new DeQueue.DeQueue()

let flag = true;

array.map( item =>{

if(item === '['){

deque.pushFront('[')

}

else if(item === ']'){

if(!deque.isEmpty()){

deque.popBack();

}

else flag = false;

}

})

return flag && !!deque.isEmpty();

}

Задание 6:

const Stack = require(`./stack`)

function regexParse (string){

let array = string.split("")

let numbers = new Stack.Stack()

let letters = new Stack.Stack()

let other = new Stack.Stack()

array.map(item => {

if (item.match(/[0-9]/)){

numbers.push(item)

}

else if (item.match(/[a-zA-Z]/)){

letters.push(item)

}

else{

other.push(item)

}

})

let numbersRevers = new Stack.Stack()

let lettersRevers = new Stack.Stack()

let otherRevers = new Stack.Stack()

while (!numbers.isEmpty()) {

numbersRevers.push(numbers.pop())

}

while (!letters.isEmpty()) {

lettersRevers.push(letters.pop())

}

while (!other.isEmpty()) {

otherRevers.push(other.pop())

}

while (!numbersRevers.isEmpty()) {

console.log(numbersRevers.pop());

}

while (!lettersRevers.isEmpty()) {

console.log(lettersRevers.pop());

}

while (!otherRevers.isEmpty()) {

console.log(otherRevers.pop());

}

}

Задание 7:

const DeQueue = require('./queue')

function numbersParse (array) {

let deque = new DeQueue.DeQueue();

array.map(item =>{

if(item < 0){

deque.pushBack(item)

}

else{

deque.pushFront(item)

}

})

while (!deque.isEmpty()){

console.log(deque.popBack())

}

}

Задание 8:

const Stack = require('./stack')

function stringRevers (string){

let array = string.split(' ')

let stack = new Stack.Stack()

array.map(string =>{

stack.push(string);

})

while(!stack.isEmpty()){

console.log(stack.pop());

}

}

Задание 9:

const Stack = require('./stack')

function computeLogic1 (Str){

let str1="";

let stk= new Stack.Stack();

for(let i=0;i<Str.length;i++){

stk.push(Str[i])

}

for(let i=0;i<Str.length;i++){

if(stk.peek()=="T")

str1+="true "

if(stk.peek()=="F")

str1+="false "

if(stk.peek()=="N")

str1+="! "

if(stk.peek()=="A" ||stk.peek()=="\*")

str1+="&& "

if(stk.peek()=="X")

str1+="!= "

if(stk.peek()=="O"||stk.peek()=="+")

str1+="|| "

if(stk.peek()=="(")

str1+="( "

if(stk.peek()==")")

str1+=")"

stk.pop()

}

console.log(eval(str1))

}

Задание 10:

const Stack = require('./stack')

function computeMinMax(Str) {

let str1=""

let stk= new Stack.Stack()

for(let i=0;i<Str.length;i++){

stk.push(Str[i])

}

for(let i=0;i<Str.length;i++){

if(stk.peek()==="0")

str1="0" +str1

if(stk.peek()==="1")

str1="1" +str1

if(stk.peek()==="2")

str1="2" +str1

if(stk.peek()==="3")

str1="3" +str1

if(stk.peek()==="4")

str1="4" +str1

if(stk.peek()==="5")

str1="5" +str1

if(stk.peek()==="6")

str1="6" +str1

if(stk.peek()==="7")

str1="7" +str1

if(stk.peek()==="8")

str1="8" +str1

if(stk.peek()==="9")

str1="9" +str1

if(stk.peek()==="M")

str1="Math.max" +str1

if(stk.peek()==="N")

str1="Math.min" +str1

if(stk.peek()===","||stk.peek()===".")

str1="," +str1

if(stk.peek()==="(")

str1="(" +str1

if(stk.peek()===")")

str1=")" +str1

stk.pop()

}

console.log(eval(str1))

}

Задание 11:

const Stack = require('./stack')

function computeForm (Str)

{

let stk= new Stack.Stack()

let str=""

for(let i=0;i<Str.length;i++){

stk.push(Str[i])

}

for (let i=0;i<Str.length;i++){

str=stk.pop()+str

}

try{

eval(str)

}

catch (err){

console.log(false)

}

console.log(true)

}

# Снимки экрана выполнения программы

Входные данные:

1: «A C B D D D SD»

2: «12315»

«)а12ко39ш(5»

3: «4, from A, inter B, to C»

4: «((((()8399())))))»

5: «[[]][]»

6: «111qqqwww11- -1q-\_+»

7: «1 -1516 71 -3»

8: «15 q1 1521 qwt»

9: «FO(TAFO(FOT))X(NT)»

10: «N(9,(M(3,N(1,2))))»

11: «x-((y+z)+(z-y))»

«x-((+z)+(z-y))»



Рис. 1 Отсортированные символы



Рис.2 Расшифрованный текст

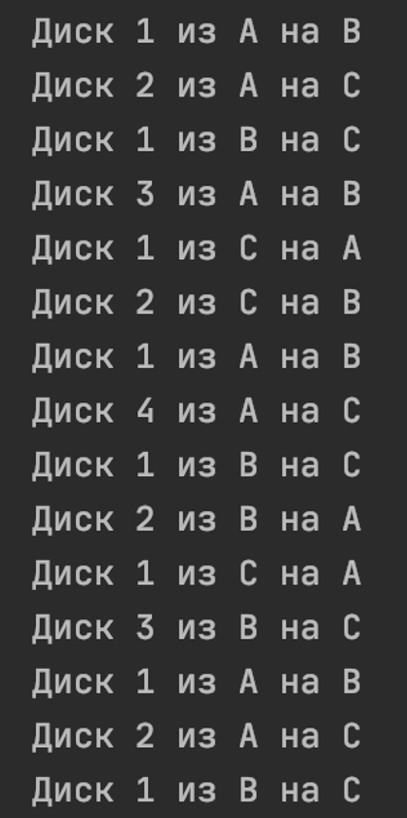


Рис.3 Выполнение задачи №3

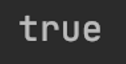


Рис.4 Выполнение задачи №4

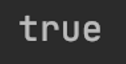


Рис.5 Выполнение задачи №5



Рис.6 Выполнение задачи №6



Рис.7 Выполнение задачи №7



Рис.8 Выполнение задачи №8



Рис.9 Выполнение задачи №9



Рис.10 Выполнение задачи №10

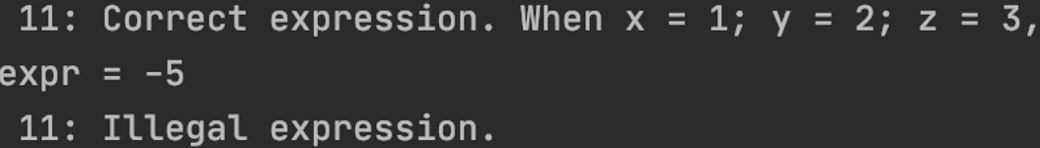


Рис.11 Выполнение задачи №11

# Вывод

В ходе выполнения лабораторной работы я реализовал структуры стек и дек, при помощи них отсортировал строку в алфавитном порядке, расшифровал сообщение, решил задачу на диски и стержни, проверил соответствие скобок, рассортировал буквы, цифры, и отрицательные, и положительные, решил логическое выражение, нашел максимум и минимум и проверил выражение.