Министерство цифрового развития, связи и массовых коммуникаций

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»


# Отчет по лабораторной работе №4

по дисциплине «Структура и алгоритмы обработки данных»

по теме «Реализация стека/дека»

Выполнил: студент группы

БВТ1902

Подпоркин В.С.

Москва

2021 г

**Оглавление**

Цель работы.....3
Задание.....3
Код программы.....5
Снимки экрана работы программ.....13
Вывод.....14

## Цель работы

Реализовать следующие структуры данных:

● 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.

## Задание 1

Отсортировать строки файла, содержащие названия книг, в алфавитном порядке с использованием двух деков.

## Задание 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.

## Задание 3

Даны три стержня и n дисков различного размера. Диски можно надевать на стержни, образуя из них башни. Перенести n дисков со стержня A на стержень C, 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 B.

Реализовать алгоритм, используя три стека вместо стержней A, B, C. Информация о дисках хранится в исходном файле.

## Задание 4

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

## Задание 5

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

## Задание 6

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

## Задание 7

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.

## Задание 8

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.

## Задание 9

Дан текстовый файл. Используя стек, вычислить значение логического выражения, записанного в текстовом файле в следующей форме: < ЛВ > ::= T | F | (N) | (A) | (X) | (O), где буквами обозначены логические константы и операции: T – True, F – False, N – Not, A – And, X – Xor, O – Or.

## Задание 10

Дан текстовый файл. В текстовом файле записана 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.

## Задание 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

# Код программы

```rust
use rand::prelude::*;
use rand::rngs::SmallRng;
use nalgebra::{U16, ArrayStorage, Dim, Matrix, Matrix2};
use std::collections::VecDeque;
use std::fmt::Display;



fn print_all<T: Display>(iter: impl IntoIterator<Item = T>) {
    for x in iter {
        print!("{} ", x)
    }
    println!();
}

const TASK8_TEXT: &str =
r"    Строчка 1
    Строчка 2
    Строчка 3
    Строчка 4";

fn main() {
    println!("Task 1, Сортировка: {:?}", &[1, 3, 5, 6, 2, 4, 3, 7, 0]);
    print!("Результат: ");
    print_all(sort_with_deck(&[1, 3, 5, 6, 2, 4, 3, 7, 0]));

    print!("Task 2, Декодирование 'анИв': ");
    // 2И16в3_а4н7
    // println!("{}", decode_message("72И14в35а6н", "713а")); // Иван
    println!("{}", decode_message("анИв", "анИв")); // Иван



    println!("Task 3, пирамидка");
    piramides(4);

    println!("Task 4, баланс скобочек:");
    println!("\t({{{}{{()}}}})[][()[]] : {}", bracket_balance("({(){()}})[][()
[]]"));
    println!("Task 5, баланс скобочек:");
    println!("\t({{{}{{()}}}})[()[]] : {}", bracket_balance("({(){()}})[()
[]]"));

    println!("Task 6, исходное \"рв+8в,42ьатш_fwnw0.3hhла2\"; Результат: {}",
task6("рв+8в,42ьатш_fwnw0.3hhла2"));
    println!("Task 7, исходное \"рв+8в,42ьатш_fwnw0.3hhла2\"; Результат: {}",
task7("рв+8в,42ьатш_fwnw0.3hhла2"));
```

```
    println!("Task 8");
    reverse_lines(TASK8_TEXT);

    print!("Task 9: Значение выражения {} = ", "FO(TAFO(FOT))X(NT)");
    task9("FO(TAFO(FOT))X(NT)"); // true

    print!("Task 10: Значение выражения {} = ", "N(9,(M(3,N(1,2))))");
    task10("N(9,(M(3,N(1,2))))"); // 3"); // true

    task11("x-((y+z)+(z-y))"); // true
    task11("x-((+z)+(z-y))"); // false

}

fn reverse_lines(source: &str) {
    let mut stack = source.lines().collect::<Vec<_>>();
    while !stack.is_empty() {
        let s = stack.pop().unwrap();
        println!("{}", s);
    }
}

fn task6(s: &str) -> String {
    let mut buf = String::new();
    let mut numbers = Vec::new();
    let mut letters = Vec::new();
    let mut other = Vec::new();

    for c in s.chars() {
        match c {
            c if c.is_digit(10) => numbers.push(c),
            c if c.is_alphabetic() => letters.push(c),
            c => other.push(c),
        }
    }

    numbers.into_iter().for_each(|x| buf.push(x));
    letters.into_iter().for_each(|x| buf.push(x));
    other.into_iter().for_each(|x| buf.push(x));

    buf.chars().collect()
}

fn task7(s: &str) -> String {
    let mut buf = String::new();
    let mut numbers = VecDeque::new();
    let mut letters = VecDeque::new();
    let mut other = VecDeque::new();
```

```
    for c in s.chars() {
        match c {
            c if c.is_digit(10) => numbers.push_back(c),
            c if c.is_alphabetic() => letters.push_back(c),
            c => other.push_back(c),
        }
    }

    numbers.into_iter().for_each(|x| buf.push(x));
    letters.into_iter().for_each(|x| buf.push(x));
    other.into_iter().for_each(|x| buf.push(x));

    buf.chars().collect()
}



fn bracket_balance(source: &str) -> bool {
    let mut stack = Vec::new();
    for c in source.chars() {
        match c {
            '(' | '[' | '{' => stack.push(c),
            ')' | ']' | '}' => match (stack.pop(), c) {
                (Some('('), ')') |
                (Some('['), ']') |
                (Some('{'), '}') => (),
                _ => return false,
            },
            _ => (),
        }
    }
    stack.len() == 0
}

fn sort_with_deck<T: Display + Ord>(source: impl IntoIterator<Item=T>) -> impl
IntoIterator<Item=T> {
    let mut source: VecDeque<T> = source.into_iter().collect();
    let mut sorted = VecDeque::with_capacity(source.len());

    while !source.is_empty() {
        let mut min = source.pop_back().unwrap();
        for i in 0..source.len() {
            let x = source.pop_back().unwrap();
            // skip_or_take_and_iter(&mut source, |x| x < min)
            //     .and_then(|x| { source.push_back(min); min = x; None});
            if min < x {
                source.push_front(min);
                min = x;
```

```
            } else {
                source.push_front(x);
            }
        }
        sorted.push_front(min);
    }
    sorted
}

fn piramides(count: usize)  {
    let mut a = Vec::with_capacity(count);
    let mut b = Vec::with_capacity(count);
    let mut c = Vec::with_capacity(count);
    for i in (0..count).rev() { a.push(i); }
    let mut n = 0;
    let mut swap_less = |da: &mut Vec<usize>, db: &mut Vec<usize>, aname: &str,
bname: &str| {
        match (da.last(), db.last()) {
            (Some(ea), Some(eb)) => {
                if ea > eb {
                    print!("\t{} -> {};", bname, aname);
                    da.push(db.pop().unwrap());
                } else {
                    print!("\t{} -> {};", aname, bname);
                    db.push(da.pop().unwrap());
                }
            }
            (Some(_), None) => {
                print!("\t{} -> {};", aname, bname);
                db.push(da.pop().unwrap())
            },
            (None, Some(_)) => {
                print!("\t{} -> {};", bname, aname);
                da.push(db.pop().unwrap())
            },
            _ => ()
        }
        if (n+1) % 5 == 0 {
            println!();
        }
        n += 1;
    };
    if count % 2 == 0 {
        while c.len() != count {
            swap_less(&mut a, &mut b, "A", "B");
            swap_less(&mut a, &mut c, "A", "C");
            swap_less(&mut b, &mut c, "B", "C");
        }
    }  else {
        while c.len() != count {
```

```
            swap_less(&mut a, &mut c, "A", "C");
            swap_less(&mut a, &mut b, "A", "B");
            swap_less(&mut b, &mut c, "B", "C");
        }
    }
}

fn skip_one<T>(deque: &mut VecDeque<T>) {
    if let Some(x) = deque.pop_back() { deque.push_front(x) };
}

fn skip_while<T>(deque: &mut VecDeque<T>, mut predicate: impl FnMut(&T) -> bool)
{
    if deque.iter().filter(|x| !predicate(*x)).count() == 0 {
        panic!("Очередь не содержит искомого элемента");
    }
    while let Some(true) = deque.get(0).map(|x| predicate(x)) {
        skip_one(deque);
    }
}

fn decode_message(rule: &str, source: &str) -> String {
    let mut rule: VecDeque<char> = rule.to_string().chars().rev().collect();
    let mut buf = String::new();

    for ch in source.chars() {
        skip_while(&mut rule, |c| *c != ch);
        skip_one(&mut rule);
        skip_one(&mut rule);
        buf.push(rule[0]);
    }
    buf
}

use parsing::*;

fn compute9(expr: &str, st: &mut Vec<bool>, deep: i32) {
    // String d = "  ".repeat(deep);
    // System.out.println(d + "EXPR: " + expr);
    if expr == "T" {
        st.push(true);
    } else if expr == "F" {
        st.push(false);
    } else if is_in_brackets(expr) {
        compute9(&expr[1..(expr.len()-1)], st, deep + 1);
    } else if expr.starts_with("N") {
        compute9(&expr[1..], st, deep + 1);
        let r = !st.pop().unwrap();
        st.push(r);
```

```
    } else if let Some(idx) = find_not_in_brackets(expr, 'O') {
        compute9(&expr[0..idx], st, deep + 1);
        compute9(&expr[(idx+1)..], st, deep + 1);
        let r = st.pop().unwrap() || st.pop().unwrap();
        st.push(r);
    } else if let Some(idx) = find_not_in_brackets(expr, 'X') {
        compute9(&expr[0..idx], st, deep + 1);
        compute9(&expr[(idx+1)..], st, deep + 1);
        let r = st.pop().unwrap() ^ st.pop().unwrap();
        st.push(r);
    } else if let Some(idx) = find_not_in_brackets(expr, 'A') {
        compute9(&expr[0..idx], st, deep + 1);
        compute9(&expr[(idx+1)..], st, deep + 1);
        let r = st.pop().unwrap() && st.pop().unwrap();
        st.push(r);
    } else {
        panic!("Illegal expression")
    //throw new Error("Illegar expression");
    }
}

fn task9(expr: &str) {
    let mut st = Vec::new();
    compute9(expr, &mut st, 0);
    let result = st.pop();
    match result {
        Some(result) => println!("Task 9: {}", result),
        None => panic!("Illegal expr.")
    }
}

fn compute10(expr: &str, st: &mut Vec<i32>, deep: i32) {
    // String d = "  ".repeat(deep);
    // System.out.println(d + "EXPR: " + expr);
    let d = "  ".repeat(deep as usize);
    //println!("{}EXPR: {}", d, expr);
    //println!("{}STCK: {:?}", d, st);

    if let Ok(x) = expr.parse::<i32>() {
        st.push(x);
    } else if is_in_brackets(expr) {
        compute10(&expr[1..(expr.len()-1)], st, deep + 1);
    } else if expr.starts_with("N") {
        compute10(&expr[1..], st, deep + 1);
        let r1 = st.pop().unwrap();
        let r2 = st.pop().unwrap();
        //println!("MIN {} {}", r1, r2);
        st.push(r1.min(r2));
    } else if expr.starts_with("M") {
```

```
        compute10(&expr[1..], st, deep + 1);
        let r1 = st.pop().unwrap();
        let r2 = st.pop().unwrap();
        //println!("MAX {} {}", r1, r2);
        st.push(r1.max(r2));
    } else if let Some(idx) = find_not_in_brackets(expr, ',') {
        compute10(&expr[0..idx], st, deep + 1);
        compute10(&expr[(idx+1)..], st, deep + 1);
    } else {
        panic!("Illegal expression")
    }

    //println!("{}STCK: {:?}", d, st);
}

fn task10(expr: &str) {
    let mut st = Vec::new();
    compute10(expr, &mut st, 0);
    let result = st.pop();
    match result {
        Some(result) => println!("Task 10: {}", result),
        None => panic!("Illegal expr.")
    }
}

fn compute11(expr: &str, st: &mut Vec<i32>, deep: i32) -> Result<(), ()>{
    // String d = "  ".repeat(deep);
    // System.out.println(d + "EXPR: " + expr);
    let d = "  ".repeat(deep as usize);
    //println!("{}EXPR: {}", d, expr);
    //println!("{}STCK: {:?}", d, st);

    if expr == "x" {
        st.push(3);
    } else if expr == "y" {
        st.push(5);
    } else if expr == "z" {
        st.push(6);
    } else if is_in_brackets(expr) {
        compute11(&expr[1..(expr.len()-1)], st, deep + 1)?;
    } else if let Some(idx) = find_not_in_brackets(expr, '-') {
        compute11(&expr[0..idx], st, deep + 1)?;
        compute11(&expr[(idx+1)..], st, deep + 1)?;
        let r1 = st.pop().ok_or(())?;
        let r2 = st.pop().ok_or(())?;
        st.push(r1 - r2);
    } else if let Some(idx) = find_not_in_brackets(expr, '+') {
        compute11(&expr[0..idx], st, deep + 1)?;
        compute11(&expr[(idx+1)..], st, deep + 1)?;
```

```
        let r1 = st.pop().ok_or(())?;
        let r2 = st.pop().ok_or(())?;
        st.push(r1 + r2);
    } else {
        return Err(());
    }

    Ok(())
    //println!("{}STCK: {:?}", d, st);
}

fn task11(expr: &str) {
    let mut st = Vec::new();
    let x = compute11(expr, &mut st, 0);
    if x.is_err() {
        println!("Task 11: Выражение {} неверно", expr);
        return;
    }
    let result = st.pop();
    match result {
        Some(result) => println!("Task 11: Выражение {} корректно", expr),
        None => println!("Task 11: Выражение {} неверно", expr),
    }
}

mod parsing {

    pub fn find_not_in_brackets(s: &str, ch: char) -> Option<usize> {
        let mut bracket = 0;
        let s_bytes = s.as_bytes();
        for i in 0..s_bytes.len() {
            match s_bytes[i] {
                c if c == b'(' => bracket += 1,
                c if c == b')' => bracket -= 1,
                c if c == (ch as u8) && bracket == 0 => return Some(i),
                _ => ()
            }
        }
        return None;
    }

    pub fn is_in_brackets(s: &str) -> bool {
        if !s.starts_with("(") || !s.ends_with(")") {
            return false;
        }
        let s = &s[1..];
        let s_bytes = s.as_bytes();
        let mut bracket = 1;
        for i in 0..s_bytes.len() {
```

```
            if bracket == 0 { return false; }
            match s_bytes[i] {
                b'(' => bracket += 1,
                b')' => bracket -= 1,
                _ =>(),
            }
            if bracket < 0 { panic!("Illegal expression"); }
        }
        return bracket == 0;
    }

}
```

## Снимки экрана работы программ

```
Task 1, Сортировка: [1, 3, 5, 6, 2, 4, 3, 7, 0]
Результат: 0 1 2 3 3 4 5 6 7
Task 2, Декодирование 'анИв': Иван
Task 3, пирамидка
    A → B; A → C; B → C; A → B; C → A;
    C → B; A → B; A → C; B → C; B → A;
    C → A; B → C; A → B; A → C; B → C;
Task 4, баланс скобочек:
    ({(){()}})[][()[]] : true
Task 5, баланс скобочек:
    ({(){()}})][()[]] : false
Task 6, исходное "рв+8в,42ьатш_fwnw0.3hhла2"; Результат: 842032рввьатшfwnwhhла+,_.
Task 7, исходное "рв+8в,42ьатш_fwnw0.3hhла2"; Результат: 842032рввьатшfwnwhhла+,_.
Task 8
    Строчка 4
    Строчка 3
    Строчка 2
    Строчка 1
Task 9: Значение выражения FO(TAFO(FOT))X(NT) = Task 9: true
Task 10: Значение выражения N(9,(M(3,N(1,2)))) = Task 10: 3
Task 11: Выражение x-((y+z)+(z-y)) корректно
Task 11: Выражение x-((+z)+(z-y)) неверно
```

Рисунок 1 – Результаты работы программы

# Вывод

Я реализовал структуры данных стек и дек и разработал алгоритмы обработки данны, в соответствии с заданиями, применяя указанную в структуру данных.