МИНОБРНАУКИ РОССИИ

Федеральное государственное
бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»
РТУ МИРЭА

Институт информационных технологий
Кафедра корпоративных информационных систем

ОТЧЕТ
по лабораторной работе №1
по дисциплине
«Структуры и алгоритмы обработки данных»

Тема лабораторной работы: «Структуры данных: список, очередь, стек»

Студент группы	ИКБО-07-18	Фроленко М.Д
Принял	ассистент кафедры КИС	Габриелян Г.А.

Выполнено «___» _____ 201__ г. _____
(подпись студента)

Зачтено «___» _____ 201__ г. _____
(подпись преподавателя)

Москва 2019

1. Задача №1

1.1. Постановка задачи

Дан текстовый файл , содержащий целые числа . Составить из него линейных однонаправленный список.

1.2. Описание используемых структур данных

- Линейный однонаправленный список – структура данных, состоящая из узлов и данных , которые в этих узлах находятся. Основу списка состовляет единственный головной узел , через который можно добраться до остальных.
- Узел – простая структура данных , содержащая в себе два поля : данные и ссылка на следующий узел.

1.3. Пользовательский интерфейс

Пользовательский интерфейс представляет из себя простое окно с тремя кнопками , одним полем ввода и вывода

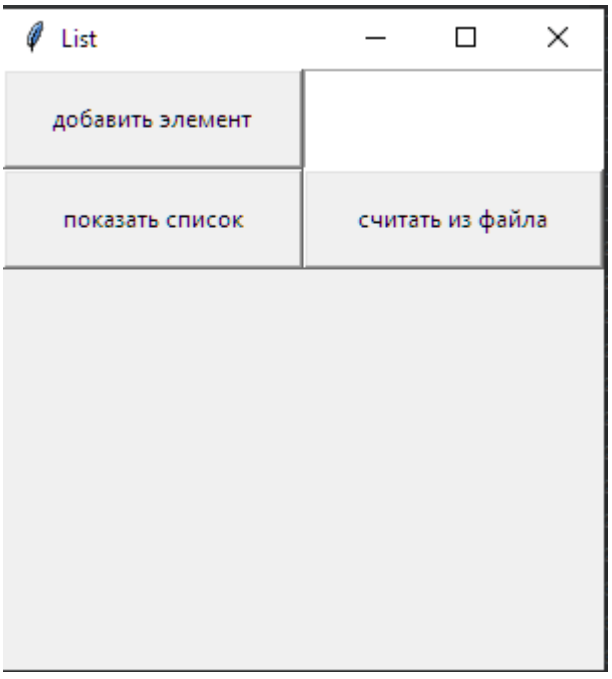

Рисунок 1 Интерфейс задания 1

1.4. Описание алгоритма

Для добавления элемента в список считывается значение из поля ввода. Для вывода списка используется обработчик на кнопку «Показать список». Из файла считывается строка , затем в цикле for заносятся элементы в

список

1.5. Тестирование

Проверена возможность добавления элемента в список – ошибки отсутсвуют. Применены различные данные внутри файла , что не сыграло на работоспособности программы. Так же список выводится спокойно

1.6. Листинг программы

1.7. List.py:

```
2. class Node:

       def __init__(self, value):
           self.value = value
           self.next = None


   class List:

       def __init__(self, node=None):
           self.head = node

       def clear(self):
           self.head = None

       def addToEnd(self, node):
           if self.head is None:
               self.head = node
           else:
               current = self.head
               while (current.next is not None):
                   current = current.next
               current.next = node

       def addToBegin(self, node):
           if self.head is None:
               self.head = node
           else:
               node.next = self.head
               self.head = node

       def remove(self, position: int):
           current = self.head
           for i in range(position):
               current = current.next
           current.next = current.next.next
```

2.1. ListWindow.py:

```
3. from tkinter import *
   from Structures import List

   mlist = List.List()
```

```
def fromFile(mlist):
    mlist.clear()
    f = open("../Files/listFile", 'r')
    for num in f.readline().split(" "):
        mlist.addToEnd(List.Node(num))



def showL():
    show.set(mlist)


def add():
    mlist.addToEnd(List.Node(entry.get()))

root = Tk()
root.title("List")
root.geometry("300x300")
show = StringVar()
entry = StringVar()


addButton = Button(root, text="добавить элемент", command= add)
addButton.place(x=0, y=0, width=150, height=50)

addEntry = Entry(root , textvariable = entry)
addEntry.place(x=150, y=0, width=150, height=50)
showButton = Button(root, text="показать список", command=  showL)
showButton.place(x=0, y=50, width=150, height=50)

showButton = Button(root, text="считать из файла" , command = lambda
: fromFile(mlist))
showButton.place(x=150, y=50, width=150, height=50)

showLabel = Label(root, textvariable=show)
showLabel.place(x=0, y=100, width=300, height=200)

root.mainloop()
```

4. Задача №2

4.1. Постановка задачи

Дан файл , содержащий вещественные числа. Составить из него очередь

4.2. Описание используемых структур данных

Очередь – структура данных , использующая принцип FIFO(First In , First Out). Имеет доступ только к начальному элементу

4.3. Пользовательский интерфейс

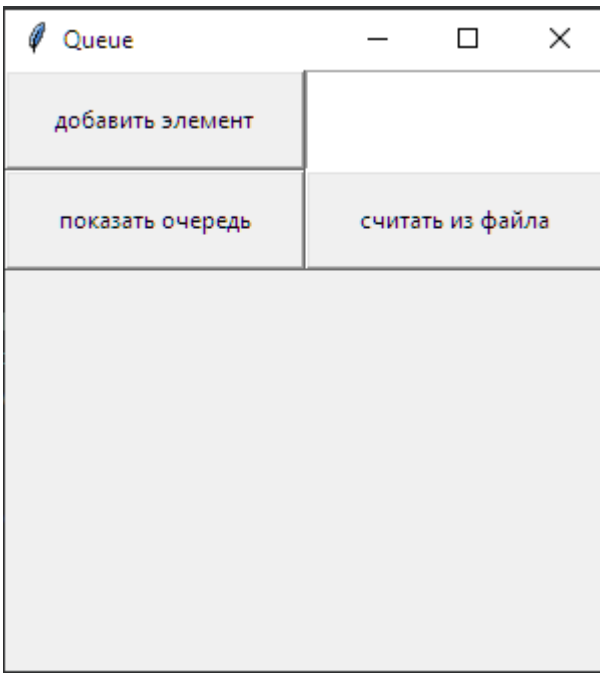

Рисунок 2 Интерфейс задания 2

4.4. Описание алгоритма

Считывается строка файла , затем разбивается по пробелам и последовательно вносится в очередь

4.5. Тестирование

Были использованы разные числа внутри файла , а так же в самом окне. Тестирование пройдено успешно

4.6. Листинг программы

4.7. Queue.py:

```
5. class Queue():

        def __init__(self, array = []):
            self.queue = array
            self.length = len(array)

        def insert(self , element):
            self.queue.append(element)

        def is_empty(self):
            return self.length == 0

        def deque(self):
            element = self.queue[0]
            self.queue.pop(0)
            return element

        def clear(self):
            self.queue = []

        def __str__(self):
            out = ""
```

```
            for e in self.queue:
                out += str(e) + " "
            return out
```

5.1. QueueWindow:

```
from tkinter import *
from Structures import Queue

root = Tk()
root.title("Queue")
root.geometry("300x300")

mqueue = Queue.Queue()

show = StringVar()
entry = StringVar()


def showL():
    show.set(mqueue)


def add():
    mqueue.insert(addEntry.get())


def fromFile(mqueue):
    mqueue.clear()
    f = open("../Files/queueFile", 'r')
    for num in f.readline().split(" "):
        mqueue.insert(float(num))

addButton = Button(root, text="добавить элемент", command=add)
addButton.place(x=0, y=0, width=150, height=50)

addEntry = Entry(root)
addEntry.place(x=150, y=0, width=150, height=50)
showButton = Button(root, text="показать очередь", command=showL)
showButton.place(x=0, y=50, width=150, height=50)

showButton = Button(root, text="считать из файла", command= lambda :
fromFile(mqueue))
showButton.place(x=150, y=50, width=150, height=50)

showLabel = Label(root, textvariable=show)
showLabel.place(x=0, y=100, width=300, height=200)

root.mainloop()
```

7. Задача №3

7.1. Постановка задачи

…Дан текстовый файл.Распечатать содержимое каждой его строки в обратном порядке , используя стек

7.2. Описание используемых структур данных

…Стек – структура данных , работающая по принципу LIFO (Last In , First Out) . Есть возможность получить доступ только к последнему элементу

7.3. Пользовательский интерфейс

…

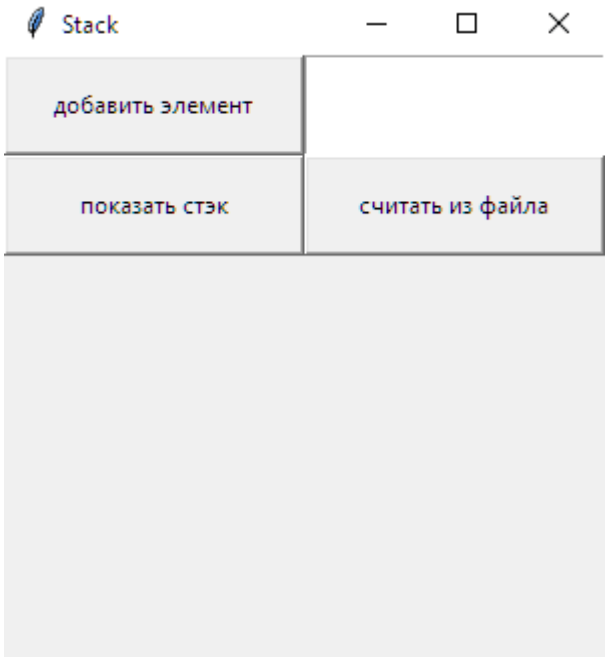

Рисунок 3 Интерфейс задания 3

7.4. Описание алгоритма

…Файл разбивается на строки . Затем каждая строка разбивается на символы , и каждый символ опускается в стек. Затем на конце строки стек очищается. Повторяется пока настигнут конец файла

7.5. Тестирование

… Были испробованы разные строки внутри файла. Тестирование пройдено успешно

7.6. Листинг программы

7.7. Stack.py:

```
8. class Stack():

        def __init__(self, array=[]):
            self.stack = array

        def pop(self):
                return self.stack.pop()

        def peek(self):
            return self.stack[-1]
```

```
        def push(self, element):
            self.stack.append(element)

        def is_empty(self):
            return self.stack == []

        def clear(self):
            self.stack = []

        def size(self):
            return len(self.stack)

        def __str__(self):
            out = ""
            for num in self.stack:
                out += str(num)+" "
            return out
```

8.1. StackWindow.py:

```
from tkinter import *
from Structures import Stack

root = Tk()
root.title("Stack")
root.geometry("300x300")

mstack = Stack.Stack()

show = StringVar()
entry = StringVar()


def add():
    mstack.push(addEntry.get())


def showS():
    show.set(mstack)




addButton = Button(root, text="добавить элемент", command=add)
addButton.place(x=0, y=0, width=150, height=50)

addEntry = Entry(root)
addEntry.place(x=150, y=0, width=150, height=50)
showButton = Button(root, text="показать стэк", command=showS)
showButton.place(x=0, y=50, width=150, height=50)

def fromFile(mstackl):
    out = ""
    mstack.clear()
    f = open("../Files/stackFile", "r")
    for line in f.readlines():
        for l in line:
            mstack.push(l)
        while mstack.size() != 0:
```

```
            out += str(mstack.pop())
        out += '\n'
    show.set(out)

showButton = Button(root, text="считать из файла", command= lambda :
fromFile(mstack))
showButton.place(x=150, y=50, width=150, height=50)

showLabel = Label(root, textvariable=show)
showLabel.place(x=0, y=100, width=300, height=200)

root.mainloop()
```