Министерство цифрового развития, связи и массовых коммуникаций

Ордена Трудового Красного Знамени федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра Математической кибернетики и информационных технологий

Отчёт по лабораторной работе №4

"Реализация стека/дека"

по дисциплине «Сиаод»

Выполнил: студент группы

БВТ1902

Сорокин Никита Андреевич

Москва

2021

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.

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 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.

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

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

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.

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.

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.

9. 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.

10. 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.

Код программы

```
class Stack:
    def __init__(self):
        self.items = []

    def isEmpty(self):
        return self.items == []

    def push(self, item):
        self.items.append(item)

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

    def show_elements(self):
        return (self.items)

    def peek(self):
        if self.items == []:
            return "error"
        else:
            return self.items[len(self.items)-1]

class Deque:
    def __init__(self):
        self.deque = []

    def pushFront(self, key):
        self.deque.insert(0, key)

    def pushBack(self, key):
        self.deque.append(key)

    def popFront(self):
        if self.deque == []:
            return "error"
        else:
            return self.deque.pop(0)

    def popBack(self):
        if self.deque == []:
            return "error"
        else:
```

```
            return self.deque.pop()

    def peekFront(self):
        if self.deque == []:
            return "error"
        else:
            return self.deque[0]

    def peekBack(self):
        if self.deque == []:
            return "error"
        else:
            return self.deque[len(self.deque)-1]

    def show_elements(self):
        return (self.deque)

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

print("")
print("Задание №1")

def bigger(s1, s2):
    s1.lower()
    s2.lower()
    for i in range (len(s1)):
        if s1[i] >s2[i]:
            return True
        if s1[i] < s2[i]:
            return False

def sort(l):
    l1 = []
    A = Deque()
    B = Deque()
    for i in range(len(l)):
        if A.deque == []:
            A.pushBack(l[i])
            continue
        if bigger(A.peekBack(), l[i]) == False:
            A.pushBack(l[i])
            continue
        if bigger(A.peekFront(), l[i]) == True:
```

```
            A.pushFront(l[i])
            continue
        x = A.peekBack()
        while bigger(x, l[i]) == True:
            x = A.popBack()
            B.pushFront(x)
            x = A.peekBack()
        A.pushBack(l[i])
        while B.deque != []:
            x = B.popFront()
            A.pushBack(x)
    while A.deque != []:
        x = A.popFront()
        l1.append(x)
    return l1

book = ["Дикие лебеди", "Дюймовочка", "Огниво", "Русалочка", "Три мушкетера",
"Баба-Яга", "Александр"]
bookSorted = sort(book)
print(bookSorted)

print()
print("Задние №2")
N2 = Deque()
str = "офхук нхщъкл дхй"
strN =
['а','б','в','г','д','е','ё','ж','з','и','й','к','л','м','н','о','п','р','с','т','у','ф','х','ц','ч','ш','щ','ъ','ы','ь','э','ю','я
']
for i in range(len(strN)):
    N2.pushBack(strN[i])


def dehash(str, D):
    str1 = ""
    for i in range(len(str)):
        if str[i] == " ":
            str1+= " "
            continue
        while len(str1)<=i:
            if D.peekBack() == str[i]:
                x = D.popBack()
                D.pushFront(x)
                x = D.popBack()
                D.pushFront(x)
```

```
            str1 += D.popBack()
            D.pushBack(str1[i])
        else:
            x = D.popBack()
            D.pushFront(x)
  return str1

print(dehash(str, N2))



list1 = [3, 2, 1, 0]
n = len(list1)
A = Stack()

for i in list1:
  A.push(i)

B = Stack()
C = Stack()


def move_disks(A, B, C, n):
  if n == 0:
     return
  elif n == 1:
     C.push(A.pop)
  elif n == 2:
     x = A.pop()
     B.push(x)

     y = A.pop()
     C.push(y)

     z = B.pop()
     C.push(z)
  else:
     move_disks(A, C, B, n - 1)
     y = A.pop()
     C.push(y)
     move_disks(B, A, C, n - 1)


move_disks(A, B, C, n)
```

```
print()
print('Задание №3')
print(A.show_elements(), B.show_elements(), C.show_elements())

print()
print("Задние №4")

str = "()(()))"
def isBalanced(str):
  N4 = Stack()
  for i in range(len(str)):
    if str[i] == "(" or str[i] == ")":
      if N4.isEmpty() == True:
        if str[i] == ")":
          return False
        N4.push(str[i])
        continue
      if str[i] == ")" and N4.peek() == "(":
        N4.pop()
        continue
      N4.push(str[i])
  if N4.isEmpty() == False:
    return False
  return True
print(isBalanced(str))

print()
print("Задание 5:")
str = "[][][]"
def isBalanced(str):
  N5 = Deque()
  for i in range(len(str)):
    if str[i] == "[" or str[i] == "]":
      if N5.size() == 0:
        if str[i] == "]":
          return False
        N5.pushBack(str[i])
        continue
      if str[i] == "]" and N5.peekBack() == "[":
        N5.popBack()
        continue
      N5.pushBack(str[i])
  if N5.size() > 0:
```

```
        return False
    return True
print(isBalanced(str))

list = ['1','2','3',"s","u",'7',"q","&",'7',"%"]

A = Stack()
B = Stack()
C = Stack()
D = Stack()

for i in range(len(list)):
    x = list[i]
    A.push(x)
    if ord(list[i]) >= 33 and ord(list[i]) <= 47:
        D.push(x)
    if ord(list[i]) >= 48 and ord(list[i]) <= 57:
        B.push(x)
    if ord(list[i]) >= 65 and ord(list[i]) <= 90:
        A.push(x)
    if ord(list[i]) >= 97 and ord(list[i]) <= 122:
        A.push(x)
    if ord(list[i]) >= 58 and ord(list[i]) <= 64:
        D.push(x)
    if ord(list[i]) >= 91 and ord(list[i]) <= 96:
        D.push(x)
    if ord(list[i]) >= 123 and ord(list[i]) <= 127:
        D.push(x)
    if ord(list[i]) >= 65 and ord(list[i]) <= 90:
        C.push(x)
    if ord(list[i]) >= 97 and ord(list[i]) <= 122:
        C.push(x)

print('')
print("Задние №6")
print(B.items + C.items + D.items)


print('')
print('Задание №7')

numbers = [1,4,23,-1,-7,52,-7,-1,2, 3]
l = []
Zad_7 = Deque()
```

```
for i in numbers:
  if i>0:
    Zad_7.pushFront(i)
    continue
  l.append(i)

x = Zad_7.peekBack()
while Zad_7.size() != 0:
  l.append(Zad_7.popBack())
  x = Zad_7.peekBack()
print(l)

print()
print("Задание №8")
A = Stack()
B = Stack()
C = Stack()

del list

str1 = "pog loki"
list = list(str1)

z = 0

for i in range(len(list)):
  A.push(list[i])

while z == 0:
  for i in range(len(A.items)):
    x = A.pop()
    if x == " ":
      break
    B.push(x)

  for i in range(len(B.items)):
    x = B.pop()
    C.push(x)

  if A.items == []:
    z = 1
    continue

  C.push(" ")
```

```
print(C.show_elements())

print()
print("Задание №9")
# задание 9
def computeLogic (str):
  str1=""
  stk= Stack()
  for i in range (len(str)):
    stk.push(str[i])
  for i in range (len(str)):
    if(stk.peek()=="T"):
      str1 +="True "
    if(stk.peek()=="F"):
      str1+="False "
    if(stk.peek()=="N"):
      str1 += "Not is "
    if(stk.peek()=="A" or stk.peek()=="*"):
      str1+="and "
    if(stk.peek()=="X"):
      str1="!= " +str1
    if(stk.peek()=="O" or stk.peek()=="+"):
      str1+="or "
    if(stk.peek()=="("):
      str1+="( "
    if(stk.peek()==")"):
      str1+=")"
    stk.pop()
  print(eval(str1))
computeLogic("TAF")

print()
print("Задание №10")
# задание 10
def computeMinMax(str):
  str1=""
  stk= Stack()
  for i in range (len(str)-1, -1, -1):
    stk.push(str[i])
  for i in range(len(str)):
    if(stk.peek()=="0"):
      str1+="0"
    if(stk.peek()=="1"):
```

```
            str1+="1"
        if(stk.peek()=="2"):
            str1+="2"
        if(stk.peek()=="3"):
            str1+="3"
        if(stk.peek()=="4"):
            str1+="4"
        if(stk.peek()=="5"):
            str1+="5"
        if(stk.peek()=="6"):
            str1+="6"
        if(stk.peek()=="7"):
            str1+="7"
        if(stk.peek()=="8"):
            str1+="8"
        if(stk.peek()=="9"):
            str1+="9"
        if(stk.peek()=="M"):
            str1+="max"
        if(stk.peek()=="N"):
            str1+="min"
        if(stk.peek()=="," or stk.peek()="."):
            str1+=","
        if(stk.peek()=="("):
            str1+="("
        if(stk.peek()==")"):
            str1+=")"
        stk.pop()
    print(eval(str1))
computeMinMax("N(3,5)")


print()
print("Задание №11")
# задание 11
def computeForm(str):
    x=1
    y=1
    z=1
    stk= Stack()
    str1=""
    for i in range(len(str) - 1, -1, -1):
        stk.push(str[i])
    for i in range(len(str)):
```

```
        str1+=stk.pop()
    try:
        eval(str1)
    except:
        print("False")
        return
    print("True")
computeForm("2 +(3+)")
```

Скриншоты работы программы

```
Задание №1
['Александр', 'Баба-Яга', 'Дикие лебеди', 'Дюймовочка', 'Огниво', 'Русалочка', 'Три мушкетера']

Задние №2
мтуси лучший вуз

Задание №3
[] [] [3, 2, 1, 0]

Задние №4
False

Задание 5:
True

Задние №6
['1', '2', '3', '7', '7', 's', 'u', 'q', '&', '%']

Задание №7
[-1, -7, -7, -1, 1, 4, 23, 52, 2, 3]

Задание №8
['l', 'o', 'k', 'i', ' ', 'p', 'o', 'g']

Задание №9
False

Задание №10
3

Задание №11
False
```

Рис. 1 – Результат выполнения заданий

Вывод

В ходе выполнения лабораторной работы, я научился реализовывать структуры данных “Стек” и “Дек”. Выполнил задания, применяя структуры на практике.