Лабораторная работа 4. Реализация стека/дека.

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.

```
import random
class LinkedNode:
    def __init__(self, value=None):
        self.value = value
        self.right = None
        self.left = None
class Stack:
    def __init__(self):
        self.head = LinkedNode()
        self.size = 0

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

    def push(self, value):
        if self.size > 0:
            node = LinkedNode(value)
            node.right = self.head
            self.head = node
        else:
            self.head.value = value
        self.size += 1

    def pop(self):
        if self.is_empty():
            return println("Стек пустой")
        remove = self.head
        if self.size > 1:
            self.head = remove.right
        self.size -= 1
        return remove.value

    def peek(self):
        if self.is_empty():
```

```
            return println("Стек пустой")
        return self.head.value

    def __len__(self):
        return self.size

    def reverse(self):
        current = self.head
        prev = None
        next = None

        while current is not None:
            next = current.right
            current.right = prev
            prev = current
            current = next

        self.head = prev
class Deque:
    def __init__(self):
        self.head = LinkedNode()
        self.tail = self.head
        self.size = 0

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

    def push_left(self, value):
        if self.size > 0:
            node = LinkedNode(value)
            node.right = self.tail
            self.tail.left = node
            self.tail = node
        else:
            self.tail.value = value
        self.size += 1

    def push(self, value):
        if self.size > 0:
            node = LinkedNode(value)
            node.left = self.head
            self.head.right = node
            self.head = node
        else:
            self.head.value = value
        self.size += 1

    def pop_left(self):
        if self.is_empty():
            return println("Стек пустой")
```

```
        remove = self.tail
        if self.size > 1:
            self.tail = remove.right
        self.size -= 1
        return remove.value

    def pop(self):
        if self.is_empty():
            return println("Стек пустой")
        remove = self.head
        if self.size > 1:
            self.head = remove.left
        self.size -= 1
        return remove.value

    def peek(self):
        if self.is_empty():
            return println("Стек пустой")
        return self.head.value

    def peek_left(self):
        if self.is_empty():
            return println("Стек пустой")
        return self.tail.value

    def __len__(self):
        return self.size
```

№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.

```
alphabet = list('абвгдеёжзийклмнопрстуфхцчшщъыьэюя')
random.shuffle(alphabet)
alphabet = ''.join(alphabet)
print(alphabet)
keyRing = Deque()
for letter in alphabet:
    keyRing.push(letter)

def encode(c):
    for i in range(len(keyRing)):
        x = keyRing.pop_left()
        if x == c:
            keyRing.push(x)
            val = keyRing.pop_left()
            keyRing.push(val)
            return val
        keyRing.push(x)
```

```
def decode(c):
    for i in range(len(keyRing)):
        x = keyRing.pop()
        if x == c:
            keyRing.push_left(x)
            val = keyRing.pop()
            keyRing.push_left(val)
            return val
        keyRing.push_left(x)

text = 'Это вторая задача'.lower()

encoded = ''
for letter in text:
    if encoded_letter := encode(letter):
        encoded += encoded_letter
    else:
        encoded += letter

print(encoded)

decoded = ''
for letter in encoded:
    if decoded_letter := decode(letter):
        decoded += decoded_letter
    else:
        decoded += letter
print(decoded)
```

```
дпиъацсюьжгрхмщклфебтзйёчшняоэуыв
узэ дзэхцо йцпцшц
это вторая задача
```

№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

```
A = Stack()
B = Stack()
C = Stack()

disks = 4

for i in range(disks, 0, -1):
    A.push(i)
```

```
def move(a, b):
    if len(a) == 0 and len(b) > 0:
        a.push(b.pop())
    elif len(a) > 0 and len(b) == 0:
        b.push(a.pop())
    elif a.peek() > b.peek():
        a.push(b.pop())
    else:
        b.push(a.pop())

if disks % 2 == 0:
    while len(C) != disks:
        move(A, B)
        move(A, C)
        move(B, C)
else:
    while len(C) != disks:
        move(A, C)
        move(A, B)
        move(B, C)

while not C.is_empty():
    print(C.pop())
```

```
1
2
3
4
```

№ 4 Дан текстовый файл с программой на алгоритмическом языке. Проверить баланс круглых скобок в тексте, используя стек.

```
def check_brackets(string):
    bracket_stack = Stack()
    for i in string:
        if i == '(':
            bracket_stack.push(i)
        elif i == ')':
            if bracket_stack.is_empty():
                return False
            bracket_stack.pop()
    return bracket_stack.is_empty()

print(check_brackets('()()()()(((('))
print(check_brackets('(()()()())'))
```

```
False
True
```

№ 5 Дан текстовый файл с программой на алгоритмическом языке. Проверить баланс квадратных скобок в тексте, используя дек.

```
def check_square_brackets(string):
    bracket_stack = Deque()
    for i in string:
```

```
        if i == '[':
            bracket_stack.push(i)
        elif i == ']':
            if bracket_stack.is_empty():
                return False
            bracket_stack.pop()
    return bracket_stack.is_empty()

print(check_square_brackets('[][[]]'))
print(check_square_brackets('[[][][]'))
```
```
True
False
```
№6 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.
```
text = '7раз отмерь, 1раз отрежь!!'

letters = Stack()
digits = Stack()
others = Stack()

for c in text:
    if c.isalpha():
        letters.push(c)
    elif c.isdigit():
        digits.push(c)
    else:
        others.push(c)

new_text = ''

letters.reverse()
digits.reverse()
others.reverse()


while not digits.is_empty():
    new_text += digits.pop()

while not letters.is_empty():
    new_text += letters.pop()

while not others.is_empty():
    new_text += others.pop()

print(new_text)
```
```
71разотмерьразотрежь ,  !!
```
№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.

```
numbers = [random.randint(-10,10)  for i in range(5)]
print(numbers)

deque = Deque()
for n in numbers:
    if n < 0:
        deque.push_left(n)
    else:
        deque.push(n)

while not deque.is_empty():
    x = deque.pop_left()
    if x < 0:
        deque.push(x)
    else:
        deque.push_left(x)
        break

while not deque.is_empty():
    x = deque.pop()
    if x < 0:
        print(x)
    else:
        deque.push(x)
        break

while not deque.is_empty():
    print(deque.pop_left())
```

```
[-5, 3, 6, -10, 4]
-5
-10
3
6
4
```

№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.

```
text = '((T)XF)O(TAT)NOT'

opstack = Stack()
vstack = Stack()

cur = 0
while True:
    read = False
    if not opstack.is_empty():
        if opstack.peek() == 'N':
            if vstack.is_empty():
```

```
            read = True
        else:
            if vstack.pop() == 'T':
                vstack.push('F')
            else:
                vstack.push('T')
            opstack.pop()
    elif opstack.peek() == 'A':
        if len(vstack) < 2:
            read = True
        else:
            a = vstack.pop()
            b = vstack.pop()
            if a == b and b == 'T':
                vstack.push('T')
            else:
                vstack.push('F')
            opstack.pop()
    elif opstack.peek() == 'O':
        if len(vstack) < 2:
            read = True
        else:
            a = vstack.pop()
            b = vstack.pop()
            if a == 'T' or b == 'T':
                vstack.push('T')
            else:
                vstack.push('F')
            opstack.pop()
    elif opstack.peek() == 'X':
        if len(vstack) < 2:
            read = True
        else:
            a = vstack.pop()
            b = vstack.pop()
            if a != b:
                vstack.push('T')
            else:
                vstack.push('F')
            opstack.pop()
    elif opstack.peek() == '(':
        read = True
    elif opstack.peek() == ')':
        opstack.pop()
        opstack.pop()
else:
    read = True
if read:
    i = text[cur]
    if i in 'FT':
```

```
            vstack.push(i)
        if i in 'AXON()':
            opstack.push(i)
        cur += 1


    if cur == len(text) and len(opstack) == 0:
        break

while not vstack.is_empty():
    print(vstack.pop())
```
T

№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.

```
text = 'N(230, N(7,M(5, 4)))'

op = Stack()
nums = Stack()

num = ''

cur = 0
while cur < len(text):
    i = text[cur]
    if i.isdigit():
        num += i
    elif num != '':
        nums.push(int(num))
        num = ''
    if i in 'MN':
        op.push(i)
    cur += 1

while not op.is_empty():
    a = nums.pop()
    b = nums.pop()
    if a < b:
        a,b = b,a
    if op.pop() == 'M':
        nums.push(a)
    else:
        nums.push(b)

while not nums.is_empty():
    print(nums.pop())
```

5

№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

```
def check(text):
    stack = Stack()

    cur = 0
    while True:
        read = False
        if not stack.is_empty():
            if stack.peek() == '(':
                read = True
            elif stack.peek() == ')':
                stack.pop()
                if len(stack) < 2 or stack.pop() != 'formula' or stack.pop() != '(':
                    return False
                stack.push('formula')
            elif stack.peek() == 'formula':
                stack.pop()
                if len(stack) > 1 and stack.peek() in '+-':
                    if stack.pop() in '+-' and stack.pop() == 'formula':
                        stack.push('formula')
                    else:
                        return False
                else:
                    stack.push('formula')
                    read = True
            else:
                read = True
        else:
            read = True
        if read:
            i = text[cur]
            if i in 'xyz':
                stack.push('formula')
            elif i in '()+-':
                stack.push(i)
            cur += 1
        if cur == len(text) and len(stack) == 1:
            break
    return True

check('((x + y) + (x - y))')
```

```
True
```

Вывод: в ходе лабораторной работы мы освоили такие структуры данных как стек и дек.