Министерство науки и высшего образования Российской Федерации

Пензенский государственный университет

Кафедра “Вычислительная техника”

**Отчет**

по лабораторной работе № 3

по курсу “Логика и основы алгоритмизации в инженерных задачах”

на тему “ Динамические списки”

Выполнили студенты группы 22ВВВ2:

Коробкин В. Ю.

Горбатов К. В.

Приняли:

Акифьев И. В.

Митрохин М. А.

Пенза 2023

**Цель работы**

Повторить и усовершенствовать полученные ранее знания о динамических списках на языке Python.

**Лабораторное задание**

1. Реализовать приоритетную очередь, путём добавления элемента в список в соответствии с приоритетом объекта (т.е. объект с большим приоритетом становится перед объектом с меньшим приоритетом).
2. На основе приведенного кода реализуйте структуру данных Очередь.
3. На основе приведенного кода реализуйте структуру данных Стек.

**Листинг**

**Lab3.1**

class Node:

def \_\_init\_\_(self, inf, priority):

self.inf = inf # полезная информация

self.priority = priority # приоритет

self.next = None # ссылка на следующий элемент

head = None # указатель на первый элемент списка

last = None # указатель на последний элемент списка

f = None # другой указатель (например, для операций)

dlinna = 0 # переменная для хранения длины списка

def get\_struct():

s = input("Введите название объекта: ") # вводим данные

priority = int(input("Введите приоритет объекта: ")) # вводим приоритет

if not s:

print("Запись не была произведена")

return None

p = Node(s, priority)

p.next = None

return p # возвращаем экземпляр созданной структуры Node

def spstore():

global head, last

p = get\_struct()

if head is None and p is not None: # если списка нет, то устанавливаем голову списка

head = p

last = p

elif head is not None and p is not None: # список уже есть, то вставляем в соответствии с приоритетом

if p.priority > head.priority: # если новый элемент имеет более высокий приоритет, то он становится первым

p.next = head

head = p

else:

current = head

while current.next is not None and current.next.priority >= p.priority:

current = current.next

p.next = current.next

current.next = p

def review():

struc = head

if head is None:

print("Список пуст")

while struc:

print(f"Имя - {struc.inf}, Приоритет - {struc.priority}")

struc = struc.next

def delete(name):

global head

struc = head # указатель, проходящий по списку установлен на начало списка

prev = None # указатель на предшествующий удаляемому элементу

flag = 0 # индикатор отсутствия удаляемого элемента в списке

if head is None: # если голова списка равна None, то список пуст

print("Список пуст")

return

if name == struc.inf: # если удаляемый элемент - первый

flag = 1

head = struc.next # устанавливаем голову на следующий элемент

struc = head # устанавливаем указатель для продолжения поиска

else:

prev = struc

struc = struc.next

while struc: # проход по списку и поиск удаляемого элемента

if name == struc.inf: # если нашли, то

flag = 1 # выставляем индикатор

if struc.next: # если найденный элемент не последний в списке

prev.next = struc.next # меняем указатели

struc = struc.next # устанавливаем указатель для продолжения поиска

else: # если найденный элемент последний в списке

prev.next = None # обнуляем указатель предшествующего элемента

return

else: # если не нашли, то

prev = struc # устанавливаем указатели для продолжения поиска

struc = struc.next

if flag == 0: # если флаг = 0, значит нужный элемент не найден

print("Элемент не найден")

loop = 0

gg = 0

while loop != 1:

print("")

print(" - - - - - - - - - - - - - - - ")

print("")

print("Выберите номер действия: ")

print("Создать структуру - 1")

print("Посмотреть список - 2")

print("Удалить элемент - 3")

print("Завершить программу - 4")

gg = int(input())

if gg == 1: # Создать структуру

spstore()

if gg == 2: # Посмотреть список

review()

if gg == 3: # Удалить элемент

print("Введите имя элемента: ")

name = input()

delete(name)

if gg == 4: # Выход из программы

loop = 1

**Lab3.2**

class Node:

def \_\_init\_\_(self, inf):

self.inf = inf # полезная информация

self.next = None # ссылка на следующий элемент

head = None # указатель на первый элемент очереди

tail = None # указатель на последний элемент очереди

dlinna = 0 # переменная для хранения длины очереди

def get\_struct():

s = input("Введите название объекта: ") # вводим данные

if not s:

print("Запись не была произведена")

return None

p = Node(s)

p.next = None

return p # возвращаем экземпляр созданной структуры Node

def enqueue():

global head, tail, dlinna

p = get\_struct()

if head is None and p is not None: # если очереди нет, то устанавливаем голову и хвост очереди

head = p

tail = p

elif head is not None and p is not None: # добавляем элемент в конец очереди

tail.next = p

tail = p

dlinna += 1 # увеличиваем длину очереди

def dequeue():

global head, dlinna

if head is None:

print("Очередь пуста")

return

removed = head

head = head.next

dlinna -= 1

return removed.inf

def review():

struc = head

if head is None:

print("Очередь пуста")

while struc:

print(f"Имя - {struc.inf}")

struc = struc.next

loop = 0

gg = 0

while loop != 1:

print("")

print(" - - - - - - - - - - - - - - - ")

print("")

print("Выберите номер действия: ")

print("Создать структуру - 1")

print("Посмотреть список - 2")

print("Удалить элемент - 3")

print("Завершить программу - 4")

gg = int(input())

if gg == 1: # Создать структуру

enqueue()

if gg == 2: # Посмотреть список

review()

if gg == 3: # Удалить элемент

dequeue()

if gg == 4: # Выход из программы

loop = 1

**Lab3.3**

class Node:

def \_\_init\_\_(self, inf):

self.inf = inf # полезная информация

self.next = None # ссылка на следующий элемент

head = None # указатель на первый элемент списка

last = None # указатель на последний элемент списка

f = None # другой указатель (например, для операций)

dlinna = 0 # переменная для хранения длины списка

top = None # указатель на вершину стека

dlinna = 0 # переменная для хранения размера стека

def get\_struct():

s = input("Введите название объекта: ") # вводим данные

if not s:

print("Запись не была произведена")

return None

p = Node(s)

p.next = None

return p # возвращаем экземпляр созданной структуры Node

def review():

struc = top

if top is None:

print("Стек пуст")

while struc:

print(f"Имя - {struc.inf}")

struc = struc.next

def push():

global top, dlinna

p = get\_struct()

if p is not None:

p.next = top # новый элемент указывает на текущую вершину стека

top = p # обновляем вершину стека

dlinna += 1 # увеличиваем размер стека

def pop():

global top, dlinna

if top is None:

print("Стек пуст")

return None

removed = top

top = top.next # обновляем вершину стека

dlinna -= 1 # уменьшаем размер стека

return removed.inf

loop = 0

gg = 0

while loop != 1:

print("")

print(" - - - - - - - - - - - - - - - ")

print("")

print("Выберите номер действия: ")

print("Создать структуру - 1")

print("Посмотреть список - 2")

print("Удалить элемент - 3")

print("Завершить программу - 4")

gg = int(input())

if gg == 1: # Создать структуру

push()

if gg == 2: # Посмотреть список

review()

if gg == 3: # Удалить элемент

pop()

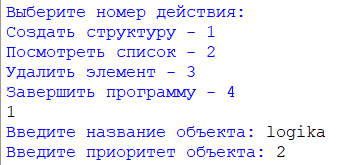
if gg == 4: # Выход из программы

loop = 1

**Результаты работы программ**

**Очередь с приоритетом**

Добавление элементов



Просмотр списка

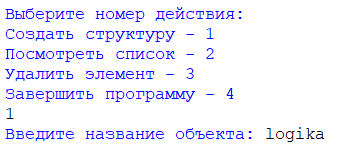


Удаление элементов

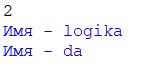


**Очередь**

Добавление элементов



Просмотр списка



Извлечение элементов из очереди



**Стек**

Добавление элементов в стек



Просмотр списка



Извлечение элементов из стека



**Вывод**

В ходе работы, удалось повторить и усовершенствовать полученные ранее знания о динамических списках на языке Python.