**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ "ЛЬВІВСЬКА ПОЛІТЕХНІКА"**

Інститут **ІКНІ**

Кафедра **ПЗ**

**ЗВІТ**

До лабораторної роботи № 8

**З дисципліни:** *“Алгоритми та структури даних”*

**На тему:** *“Лінійні структури даних”*

**Лектор:**

доц. каф. ПЗ

Коротєєва Т.О.

**Виконав:**

ст. гр. ПЗ – 22

Солтисюк Д.А.

**Прийняв:**

асист. каф. ПЗ

Франко А.В.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2022 р.

∑= \_\_\_\_\_ .

Львів – 2022

**Тема роботи:** Лінійні структури даних.

**Мета роботи:** познайомитися з лінійними структурами даних (стек, черга, дек, список) та отримати навички програмування алгоритмів, що їх обробляють.

**ТЕОРЕТИЧНІ ВІДОМОСТІ**

Стек, черга, дек, список відносяться до класу лінійних динамічних структур.

Зі стеку (stack) можна видалити тільки той елемент, який був у нього доданий останнім: стек працює за принципом «останнім прийшов – першим пішов» (last-in, first-out – LIFO).

З черги (queue), навпаки, можна видалити тільки той елемент, який знаходився в черзі довше за всіх: працює принцип «першим прийшов – першим пішов» (first-in, first-out – FIFO).

Дек - це впорядкована лінійна динамічно змінювана послідовність елементів, у якій виконуються такі умови: 1) новий елемент може приєднуватися з обох боків послідовності; 2) вибірка елементів можлива також з обох боків послідовності. Дек називають реверсивною чергою або чергою з двома боками.

У зв’язаному списку (або просто списку; linked list) елементи лінійно впорядковані, але порядок визначається не номерами, як у масиві, а вказівниками, що входять до складу елементів списку. Списки є зручним способом реалізації динамічних множин.

Елемент двобічно зв’язаного списку (doubly linked list) – це запис, що містить три поля: key (ключ) і два вказівники next (наступний) і prev (попередній). Крім цього, елементи списку можуть містити додаткові дані.

У кільцевому списку (circular list) поле prev голови списку вказує на хвіст списку, а поле next хвоста списку вказує на голову списку.

**ІНДИВІДУАЛЬНЕ ЗАВДАННЯ**

Розробити програму, яка читає з клавіатури послідовність даних, жодне з яких не повторюється, зберігає їх до структури даних (згідно з варіантом) та видає на екран такі характеристики:

* кількість елементів;
* мінімальний та максимальний елемент (для символів за кодом);
* третій елемент з початку послідовності та другий з кінця послідовності;
* елемент, що стоїть перед мінімальним елементом та елемент, що стоїть після максимального;
* знайти позицію елемента, значення якого задається з клавіатури;
* об'єднати дві структури в одну.

Всі характеристики потрібно визначити із заповненої структури даних.  
Використовувати готові реалізації структур даних (наприклад, STL) заборонено.

Варіант 22 (7), черга дійсних

**ВИКОНАННЯ РОБОТИ**

**Код програми:**

Файл queue.py:

**class** Queue:

**def** \_\_init\_\_(self):

self.items = []

*# Add an element*

**def** enqueue(self, item):

self.items.append(item)

*# Remove an element*

**def** dequeue(self):

**if** len(self.items) < 1:

**return None**

**return** self.items.pop(0)

*# Merge queues*

**def** merge(self, other\_queue):

self.items = self.items + list(set(other\_queue.items) - set(self.items))

**return** self

**def** \_\_len\_\_(self):

**return** len(self.items)

Файл main.py:

**import** pprint

**import** random

**from** textwrap **import** dedent

**from** lab8.queue **import** Queue

**def** gen\_input\_data(n):

low, high = 1, 1000

**return** random.sample(range(low, high), n)

**def** show\_statistics(q):

min\_el = max\_el = q.items[0]

next\_max\_el = **None**

pre\_min\_el = **None**

second\_from\_end = third\_el = **None**

n = len(q)

**for** idx, el **in** enumerate(q.items):

**if** idx == 2:

third\_el = el

**if** idx == n - 2:

second\_from\_end = el

**if** min\_el > el:

min\_el = el

pre\_min\_el = q.items[idx - 1]

**if** max\_el < el:

max\_el = el

next\_max\_el = q.items[idx + 1]

**print**(

dedent(

f"""

Elements count: {n}

Min element: {min\_el}

Max element: {max\_el}

3rd element from the start: {third\_el}

2nd element from the end: {second\_from\_end}

Pre-min element: {pre\_min\_el}

Post-max element: {next\_max\_el}

"""

)

)

**def** populate\_queue(n):

q = Queue()

data = gen\_input\_data(n)

**for** char **in** data:

q.enqueue(char)

**print**("\nCreated Queue on array implementation\n")

pp = pprint.PrettyPrinter(width=60, compact=**True**)

**print**(f"Data acquired:\n {pp.pformat(data)}\n")

**return** q

**def** main():

q1 = populate\_queue(100)

q2 = populate\_queue(200)

**print**("Statistics:")

show\_statistics(q1)

q1.merge(q2)

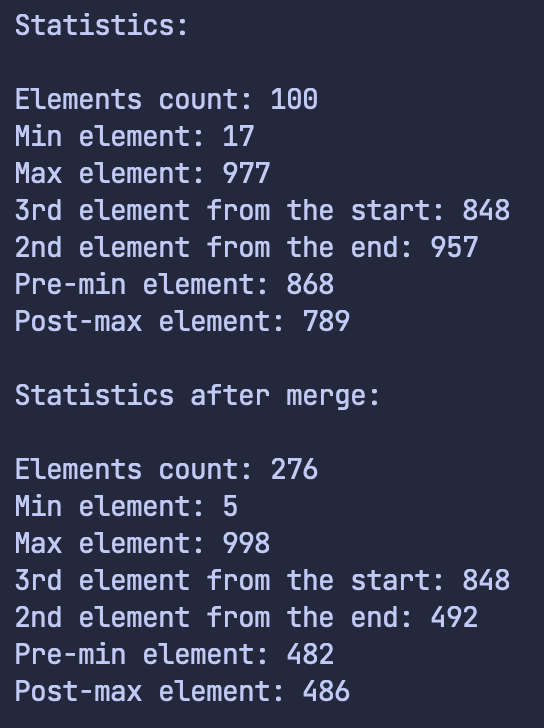
**print**("Statistics after merge:")

show\_statistics(q1)

main()

**ПРОТОКОЛ РОБОТИ**

Результатом запуску програми є стандартна інформація про задану чергу, що вимагається в пунктах до індивідуального завдання:



**ВИСНОВКИ**

У цій лабораторній роботі я ознайомився з лінійною структурою даних “черга” та отримав навички програмування алгоритмів, що її обробляють.