МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний аерокосмічний університет ім. М.Є. Жуковського

«Харківський авіаційний інститут»

Факультет радіоелектроніки, комп’ютерних систем та інфокомунікацій

Кафедра комп’ютерних систем, мереж і кібербезпеки (503)

Лабораторна робота № *3*

|  |  |
| --- | --- |
|  | *Дослідження організації спискових структур* |
|  | (назва лабораторної роботи) |
| з дисципліни | *Моделі та структури даних* |
|  | (шифр)  ХАІ**.**503**.**525a**.**03О**.**123-Комп'ютерна інженерія**,** ПЗ №9629619 |

|  |  |  |
| --- | --- | --- |
| Виконав студент гр. | 525а | *Литвиненко А.В.* |
|  | (№ групи) | (П.І.Б.) |
| (підпис, дата) |  |  |
| Перевірив | канд. техн. наук, доцент | |
|  |  | *А. В. Шостак* |
| (підпис, дата) |  | (П.І.Б.) |

Харків – 2022

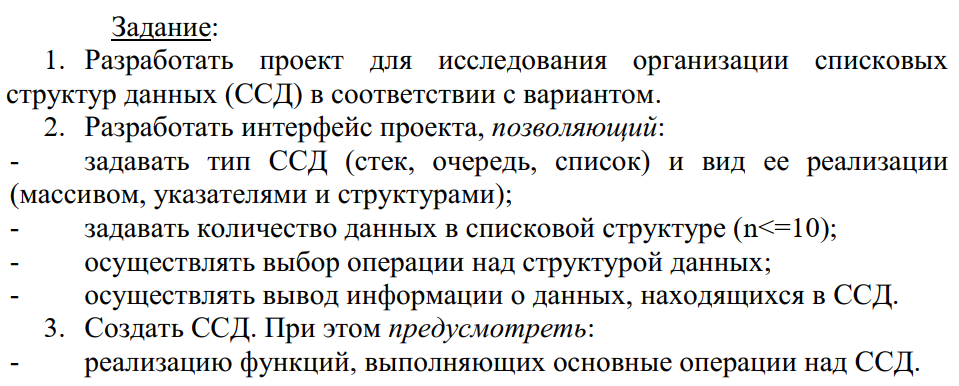
**Тема роботи:** дослідження організації спискових структур даних

**Варіант 5**

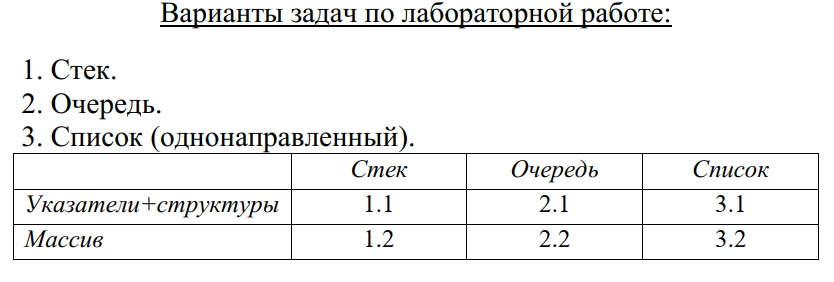
**Задача 1**

**Частина 1**. Постановка завдання

**Умова:**



**Умова з додатка:**





**Частина 2**. Схема класу

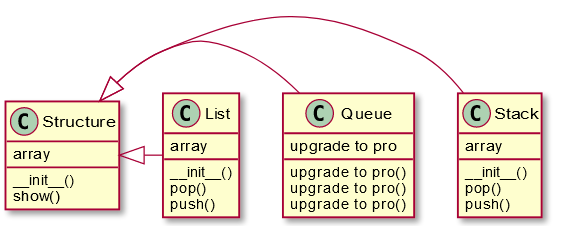


Рисунок 1 - Схема

**Частина 3**. Текст програми

class Structure:

    def \_\_init\_\_(self):

        self.array = []

        self.length = 0

    def show(self):

        for i in range(len(self.array)-1, -1, -1):

            print(f"{i+1}. {self.array[i]}")

        print()

class Stack(Structure):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

    def push(self, arg):

        self.array.append(arg)

        self.length += 1

    def pop(self):

        self.array.pop()

        self.length -= 1

class Queue(Structure):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

    def push(self, arg):

        self.array.append(arg)

        self.length += 1

    def pop(self):

        self.array.pop(0)

        self.length -= 1

class List(Structure):

    def \_\_init\_\_(self):

        super().\_\_init\_\_()

    def push(self, arg, index: int = 0):

        self.array.insert(index, arg)

        self.length += 1

    def pop(self, index: int = 0):

        self.array.pop(index)

        self.length -= 1

def menu() -> None:

    a = """Menu:

    - 0. Show this menu

    - 1. Set array len

    - 2. Set structure

    - 3. Push (insert into)

    - 4. Pop  (delete from)

    - 5. Show

    - 6. Exit"""

    print(a)

op = 0

# Length

l = 1

main = None

while 1:

    match op:

        # Show menu

        case 0:

            menu()

        # Set array len

        case 1:

            l = int(input('[LENGTH] >> '))

        # Set structure

        case 2:

            print('[ SELECT STRUCTURE ]')

            print('- 1. Stack\n- 2. Queue\n- 3. List')

            ss = int(input('[STRUCT] >> '))

            match ss:

                case 1:

                    main = Stack()

                case 2:

                    main = Queue()

                case 3:

                    main = List()

                case \_:

                    print('[ERR] Invalid operation')

            if main is not None:

                main.length = l

            print('[OK]')

        # Push

        case 3:

            if main is None:

                op = 2

                continue

            if isinstance(main, (Stack, Queue)):

                arg = input('[ARGUMENT] >> ')

                main.push(arg)

            else:

                arg, idx = input('[ARGUMENT INDEX] >> ').split(' ')

                idx = int(idx)

                main.push(arg, idx)

            print('[OK]')

        # Pop

        case 4:

            if main is None:

                op = 2

                continue

            if isinstance(main, (Stack, Queue)):

                main.pop()

            else:

                idx = int(input('[INDEX] >> '))

                main.pop(idx)

            print('[OK]')

        # Show

        case 5:

            if main is None:

                op = 2

                continue

            main.show()

            pass

        # Exit

        case 6:

            print("[OK] Exiting...")

            exit()

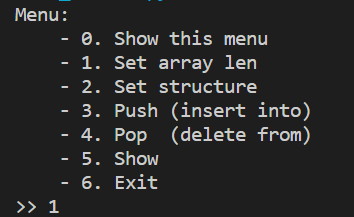
        # Else

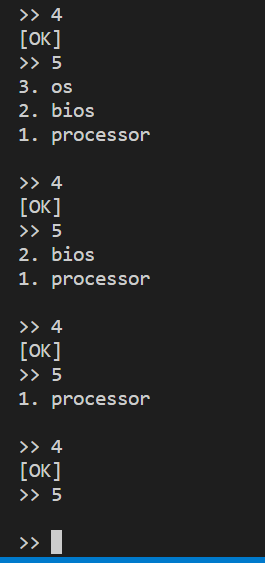
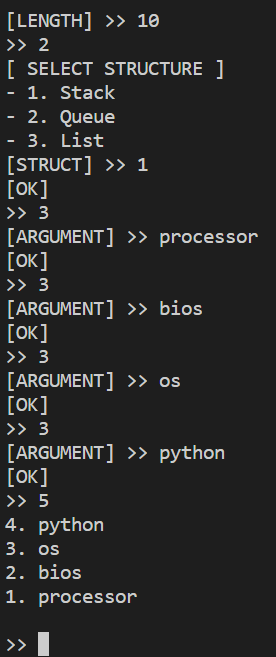
        case \_:

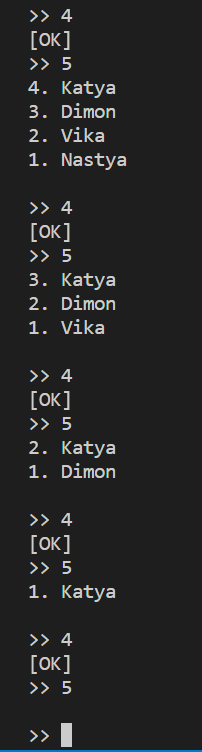
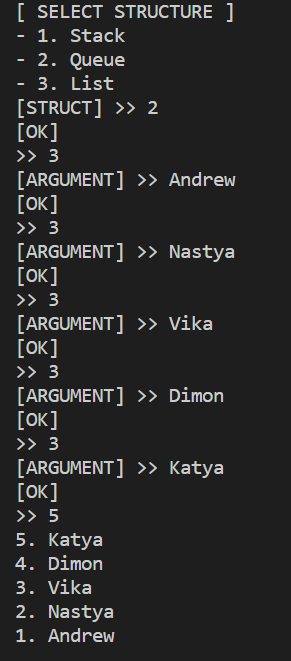
            print("[ERR] An invalid operation!")

    op = int(input('>> '))

**Частина 4**. Результат виконання







**Висновки**

У своєму до цієї лабораторної роботи у якій я вивчав різні спискові структури даних я мову відмітити, що стек дуже добре підходить для систем де важлива послідовність, тобто, що без процесора не запуститься біос, а без біоса не буде працювати операційна система, тобто це нагадує різні рівні абстракції, працює шляхом «перший прийшов – останнім вийшов».

Черга ж більше підходить для послідовного обслуговування, як у лікарні: доктор – це процесор, а клієнти – задачі; він послідовно обслуговує кожного клієнта шляхом перший прийшов – перший отримав відповідь.