Практическая работа №4.

Тема: «Структуры данных «связный список».

Цель работы: изучить СД типа «связный список», научиться их программно реализовывать и использовать.

Ход работы:

Для реализации линейного списка сначала определим структуру данных для узла этого списка. Класс Node представлен на рисунке 1.

```
class Node:
    def __init__(self, data):
        self.data = data
        self.next = None

    def get_data(self):
        return self.data

    def get_next(self):
        return self.next

    def set_next(self, next):
        self.next = next
```

Рис.1 Класс Node

Функция для добавления элемента в начало списка представлена на рисунке 2.

```
def push(self, value):
    temp = Node(value)
    temp.set_next(self.head)
    self.head = temp
```

Рис.2 Функция для добавления элемента в начало списка

Функция для вставки элемента в конец списка представлена на рисунке 3.

```
def append(self, new_data):
    new_node = Node(new_data)
    if self.head is None:
        self.head = new_node
        return
    last = self.head
    while last.next:
        last = last.get_next()
    last.set_next(new_node)
```

Рис. 3 Функция для добавления элемента в конец списка

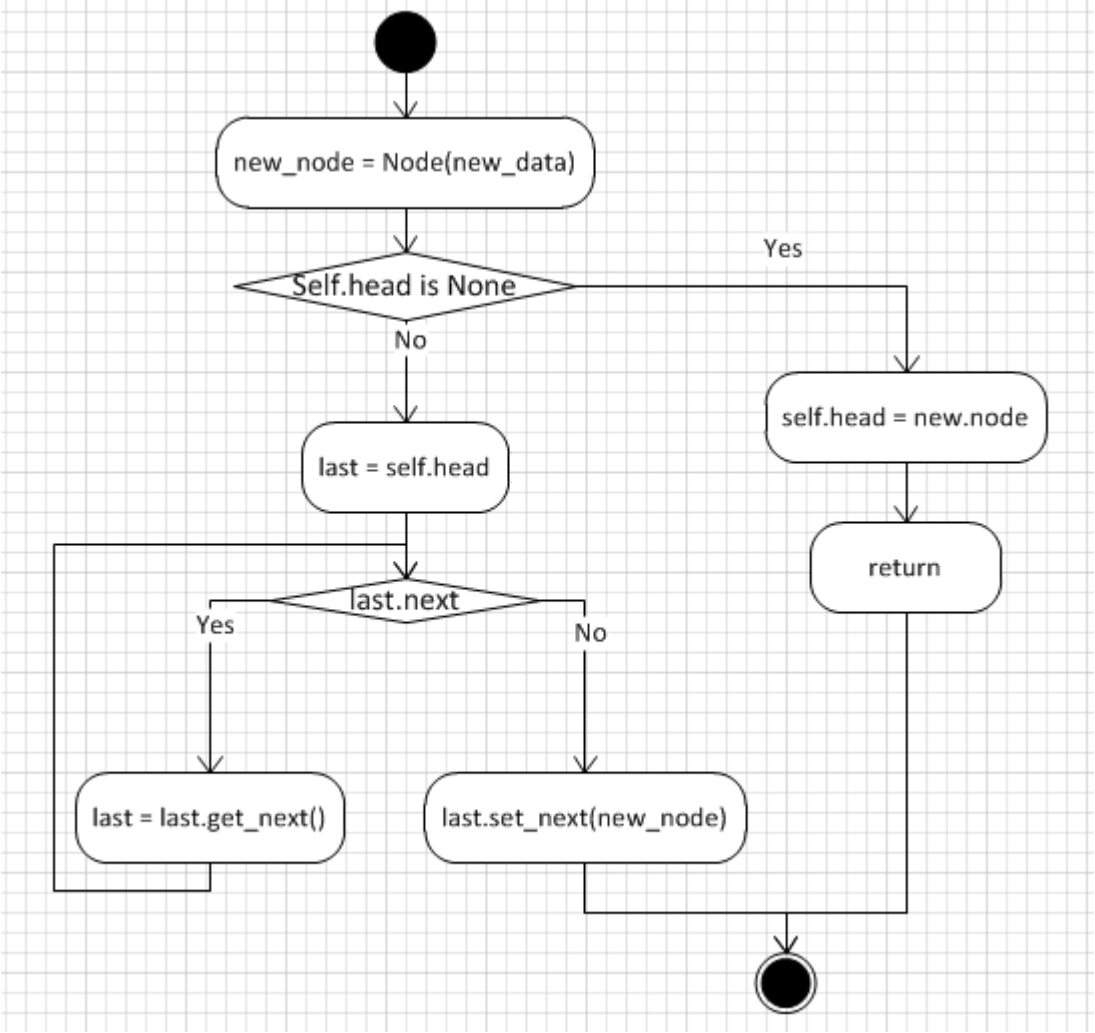

Рис 4. Диаграмма деятельностей для функции добавления элемента в конец списка.

Функция для добавления элемента в произвольное место списка представлена на рисунке 5.

```
def insert_after(self, key, new_data):
    current = self.head
    found = False
    while current is not None and not found:
        if current.get_data() == key:
            found = True
        else:
            current = current.get_next()
    if found:
        new_node = Node(new_data)
        new_node.set_next(current.get_next())
        current.set_next(new_node)
```

Рис. 5 Функция добавления элемента в произвольное место списка

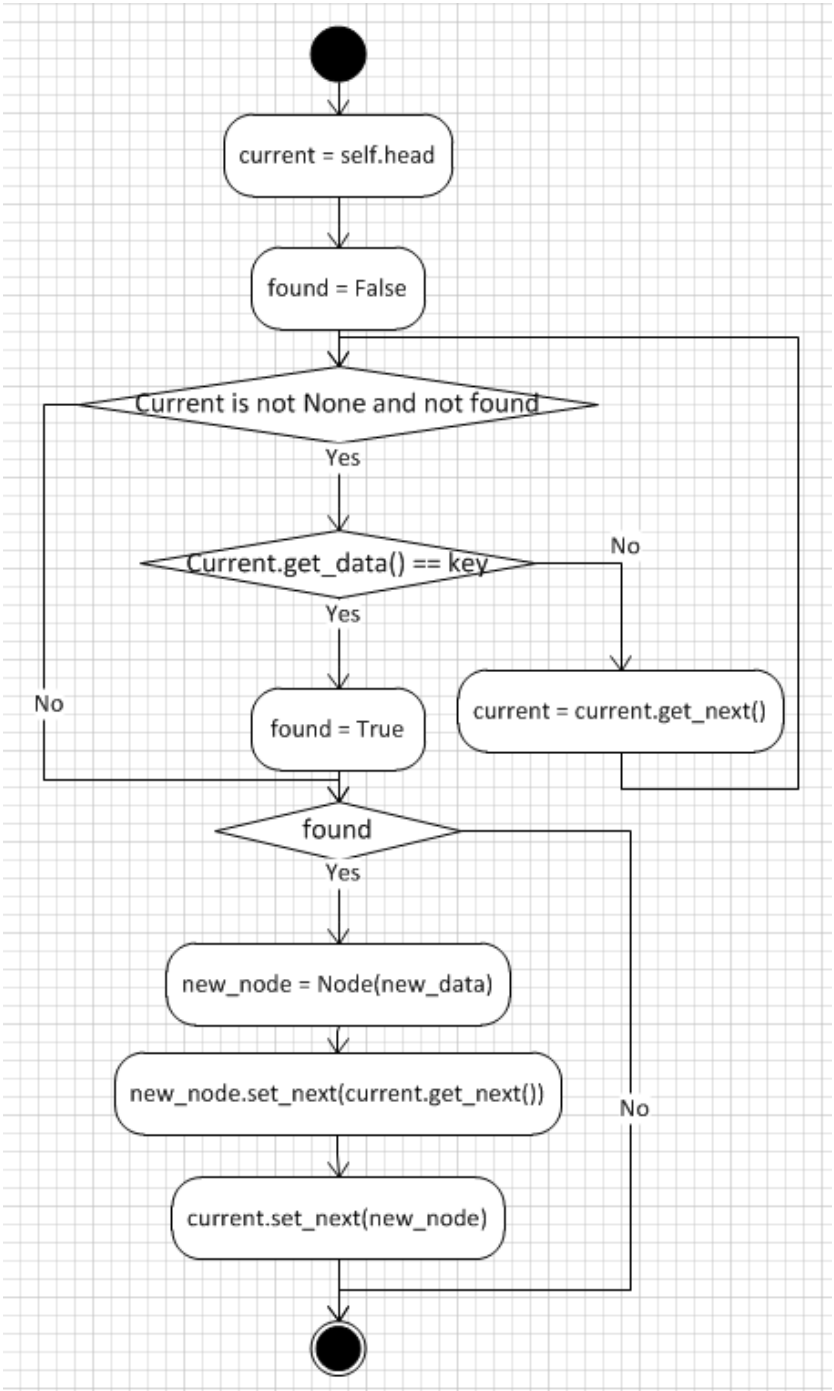

Рис. 6 Диаграмма деятельностей для функции добавления элемента в произвольное место списка.

Функция для поиска элемента в списке представлена на рисунке 7.

```
def search(self, key):
    current = self.head
    found = False
    while current is not None and not found:
        if current.get_data() == key:
            found = True
        else :
            current = current.get_next()
    return found
```

Рис. 7 Функция поиска элемента в списке

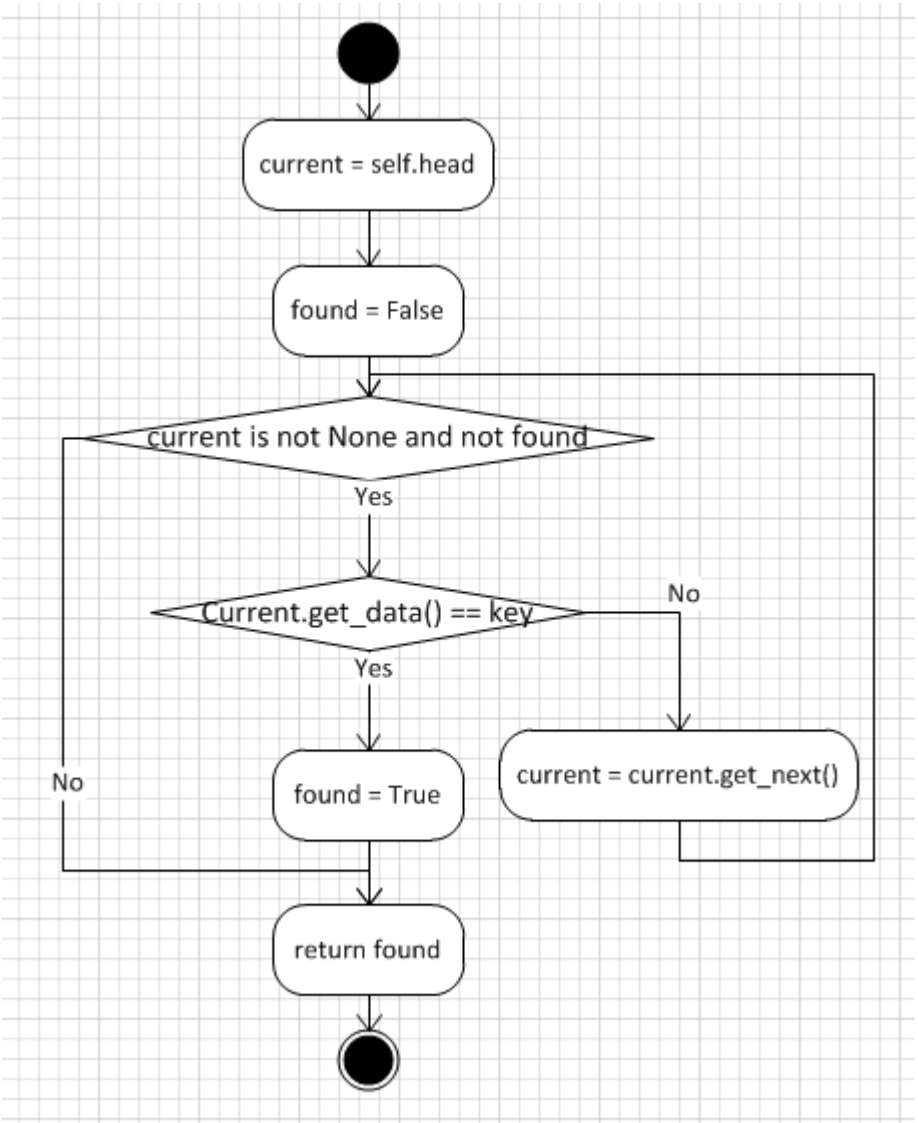

Рис. 8 Диаграмма деятельностей для функции поиска элемента в списке.

Функция для удаления элемента списка представлена на рисунке 9.

```
def delete_node(self, key):
    current = self.head
    previous = None
    found = False
    while not found:
        if current.get_data() == key:
            found = True
        else:
            previous = current
            current = current.get_next()
    if previous is None:
        self.head = current.get_next()
    else:
        previous.set_next(current.get_next())
```

Рис. 9 Функция для удаления элемента списка.

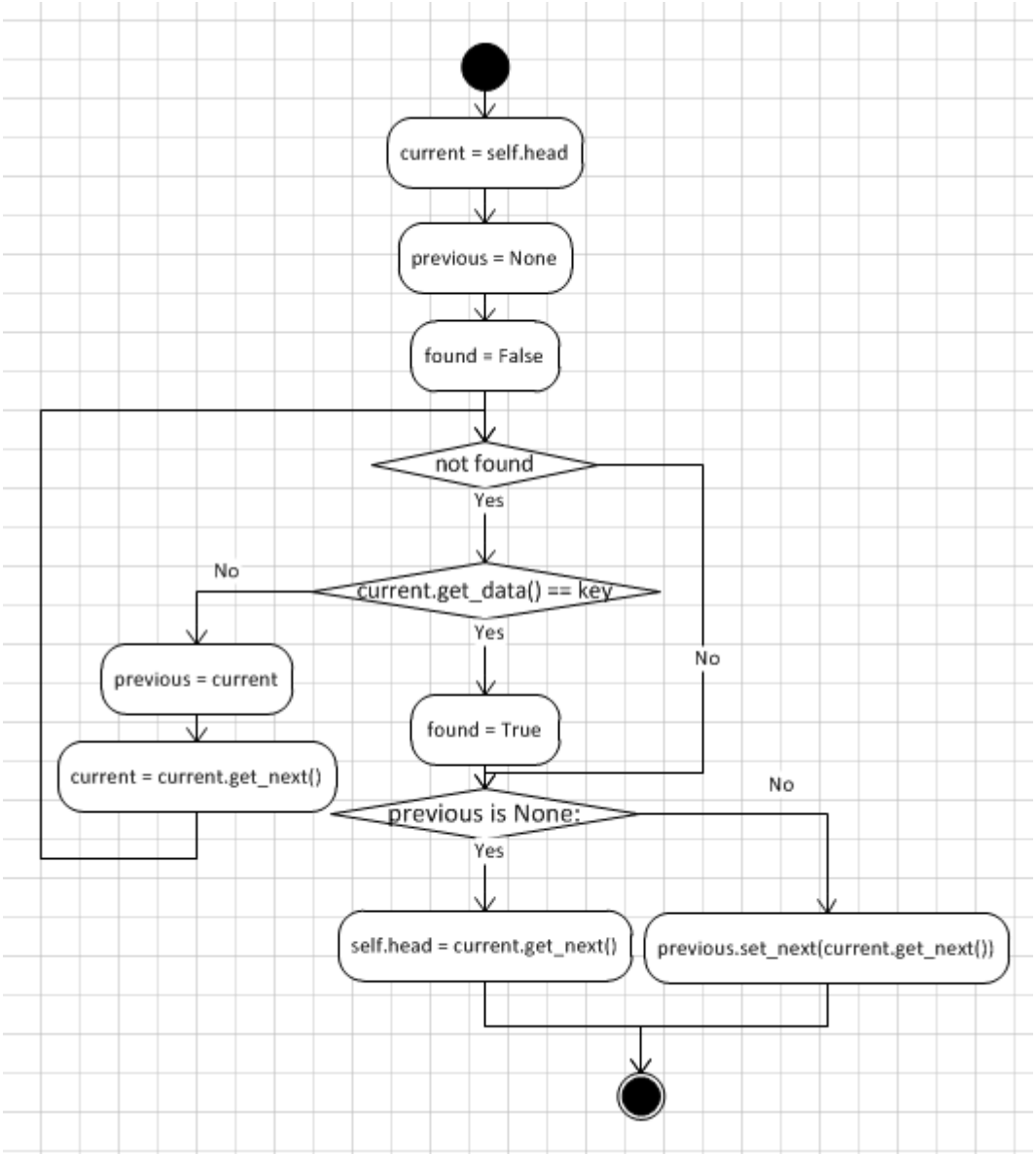

Рис. 10 Диаграмма деятельностей для функции удаления элемента списка.

Вывод: в ходе выполнения данной практической работы была реализована структура данных связный список.