Практическая работа №4

Тема: «Связанный список».

Цель работы: изучить связанные списки, научиться их программно реализовывать и использовать.

Для реализации «линейного списка» сначала необходимо определить структуру узла, код которого представлен на Рис. 1.

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

Рис. 1. Структура узла.

Код метода вставки элемента в произвольное место представлен ниже.

```
def insert(self, index, value):
    if self.first is None:
        self.first = Node(value, self.first)
        self.last = self.first.next
        return
    if index == 0:
        self.push(value)
        return
    current = self.first
    count = 0
    while current is not None:
        if count == index - 1:
            current.next = Node(value, current.next)
            if current.next is None:
                self.last = current.next
            break
        current = current.next
        count += 1
```

Рис. 2. Код вставки элемента.

Диаграмма деятельности для вставки элемента представлена на Рис. 3.

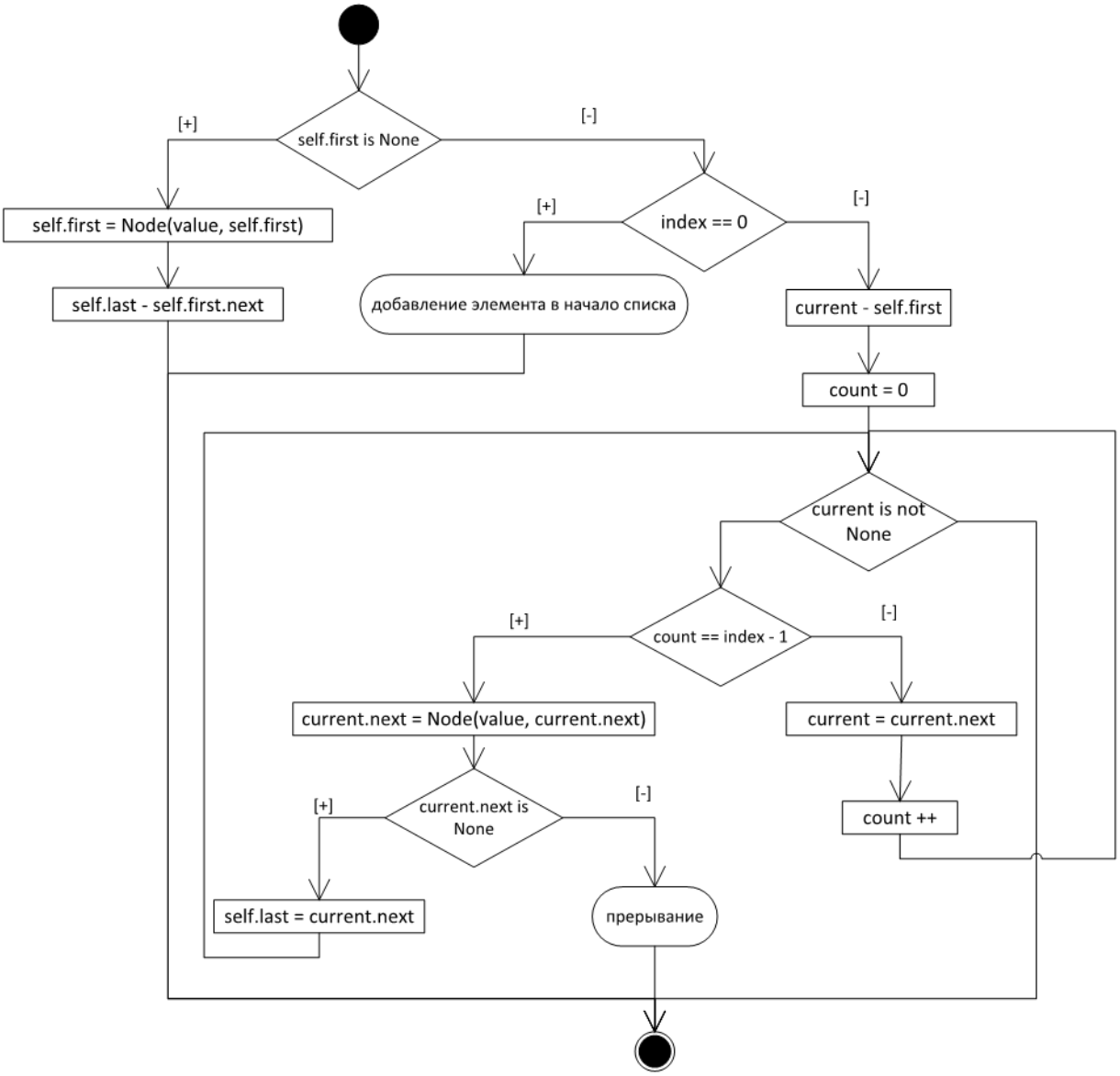

Рис. 3. Диаграмма деятельности для вставвки элемента в произвольную позицию в списке.

Код метода добавления элемента в конец списка представлен на Рис. 4. Диаграмма деятельности для этого метода приведена на Рис. 5.

```
def clear(self):
    self.__init__()

def add(self, x):
    self.length += 1
    if self.first is None:
        self.first = Node(x, None)
        self.last = self.first
    else:
        node = Node(x, None)
        self.last.next = node
        self.last = node
```

Рис. 4. Реализация метода добавления элемента в конец списка.

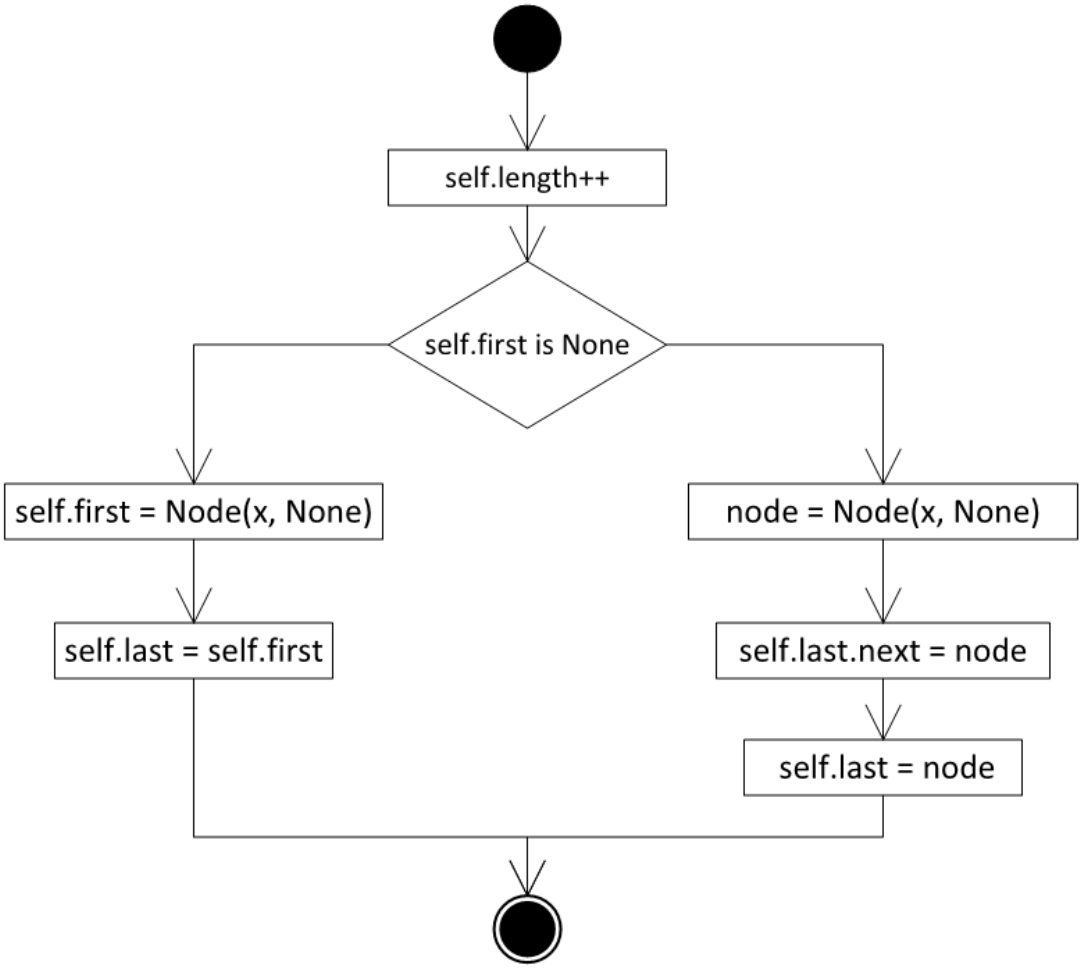

Рис. 5. Добавление элемента в конец списка.

Метод добавления в начало списка представлен на Рис. 6, а диаграмма деятельности для него приведена на Рис. 7.

```python
def push(self, x):
    self.length += 1
    if self.first is None:
        self.first = Node(x, None)
        self.last = self.first
    else:
        self.first = Node(x, self.first)
```

Рис. 6. Метод добавления в начало списка.

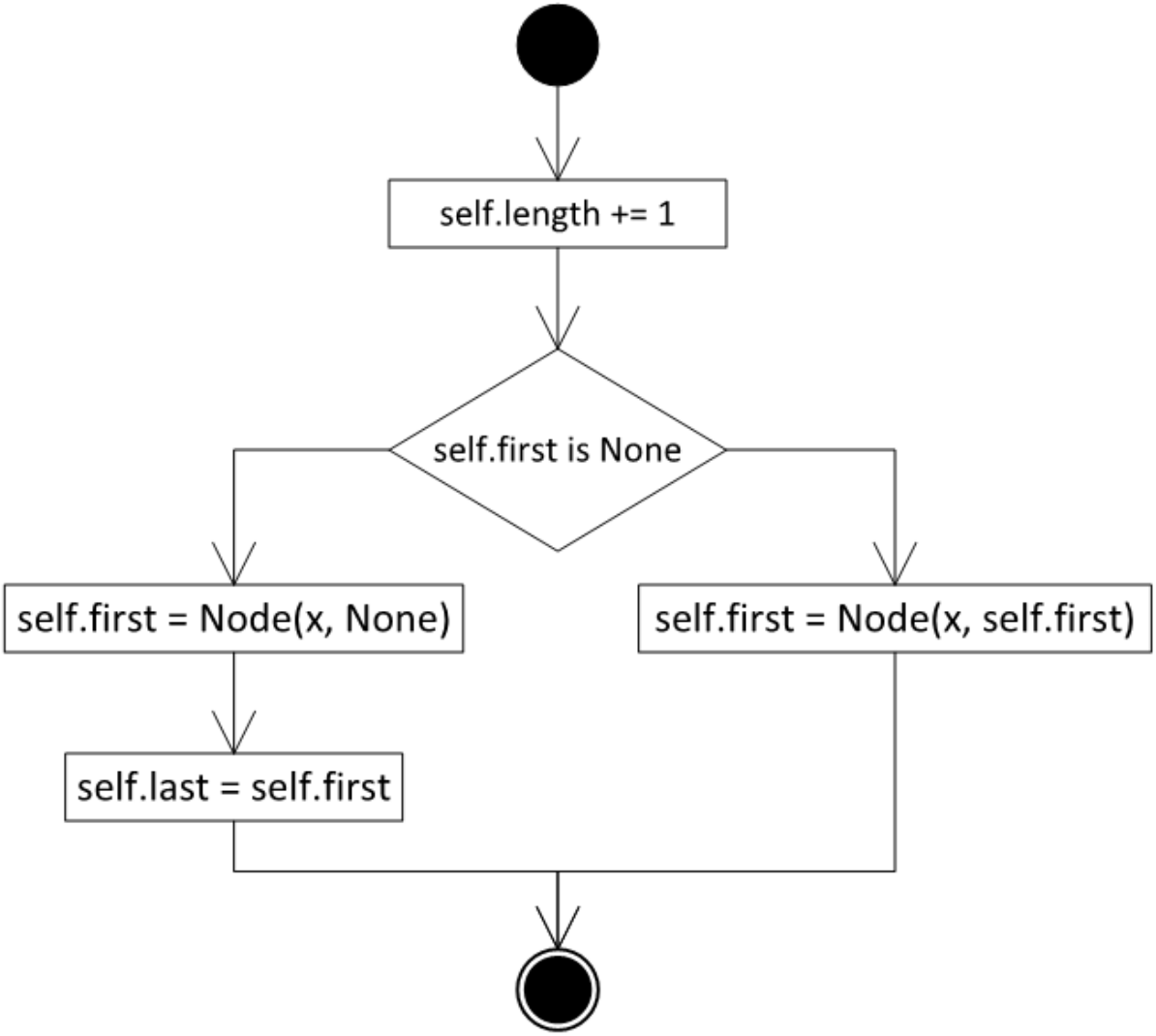

Рис. 7. Добавление элемента в начало списка.

Код для удаления головного элемента списка приведен на Рис. 8. Диаграмма деятельности для него представлена на Рис. 9.

```
def pop(self):
    oldhead = self.first
    if oldhead is None:
        return None
    self.first = oldhead.next
    if self.first is None:
        self.last = None
    return oldhead.value
```

Рис. 8. Удаление головного элемента.

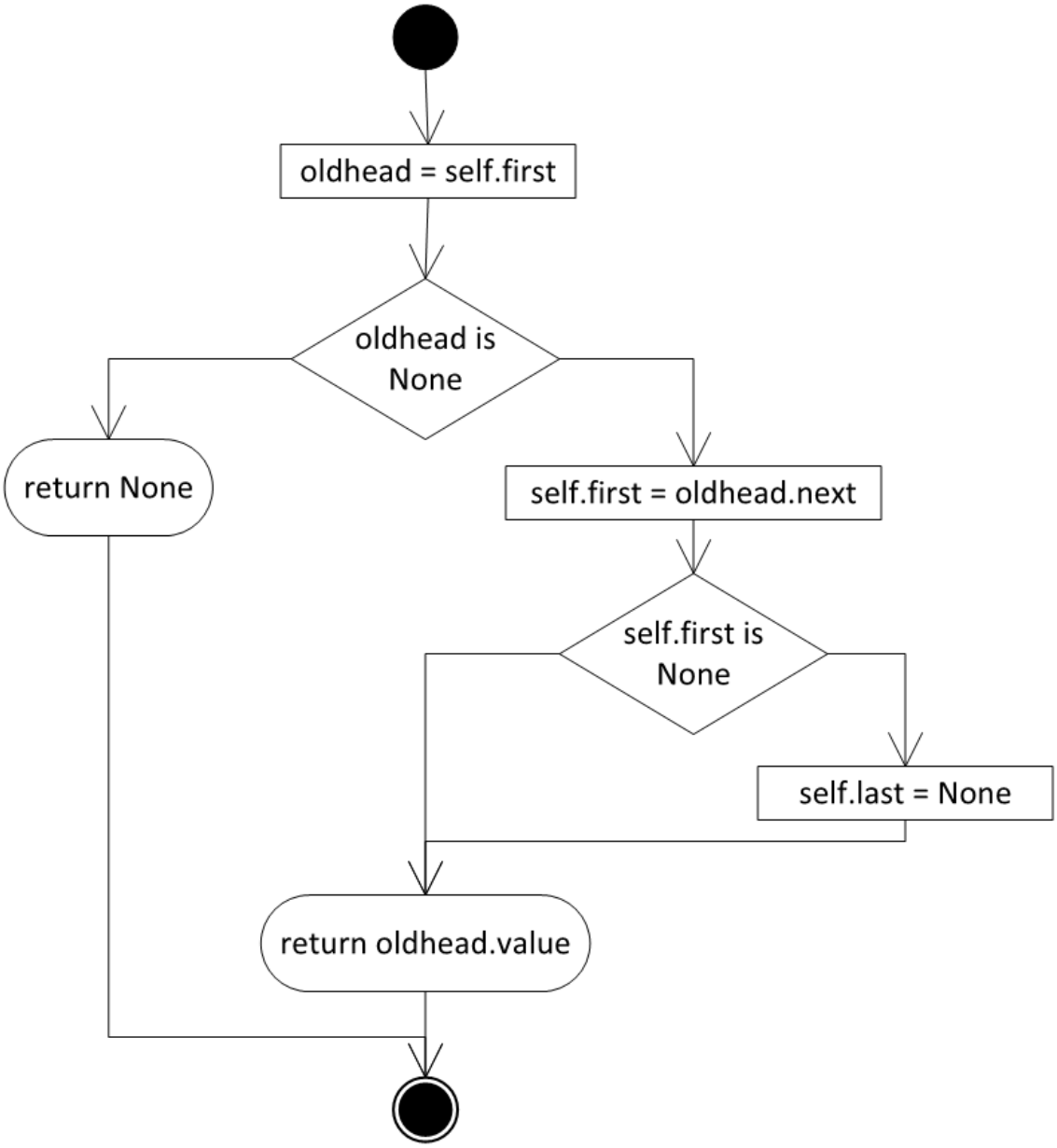

Рис. 9. Удаление головного элемента.

Метод удаления элемента по его значению представлен на Рис. 10. Диаграмма деятельности для него представлен на Рис. 11.

```
def del_element(self, value):
    first = self.first
    if first is not None and first.value == value:
        self.first = first.next
        first = None
        self.length -= 1
        return
    while first is not None or value != first.value:
        last = first
        first = first.next
    if first is None:
        return
    last.next = first.next
    first = None
    self.length -= 1
```

Рис. 10. Удаление элемента по его значению.

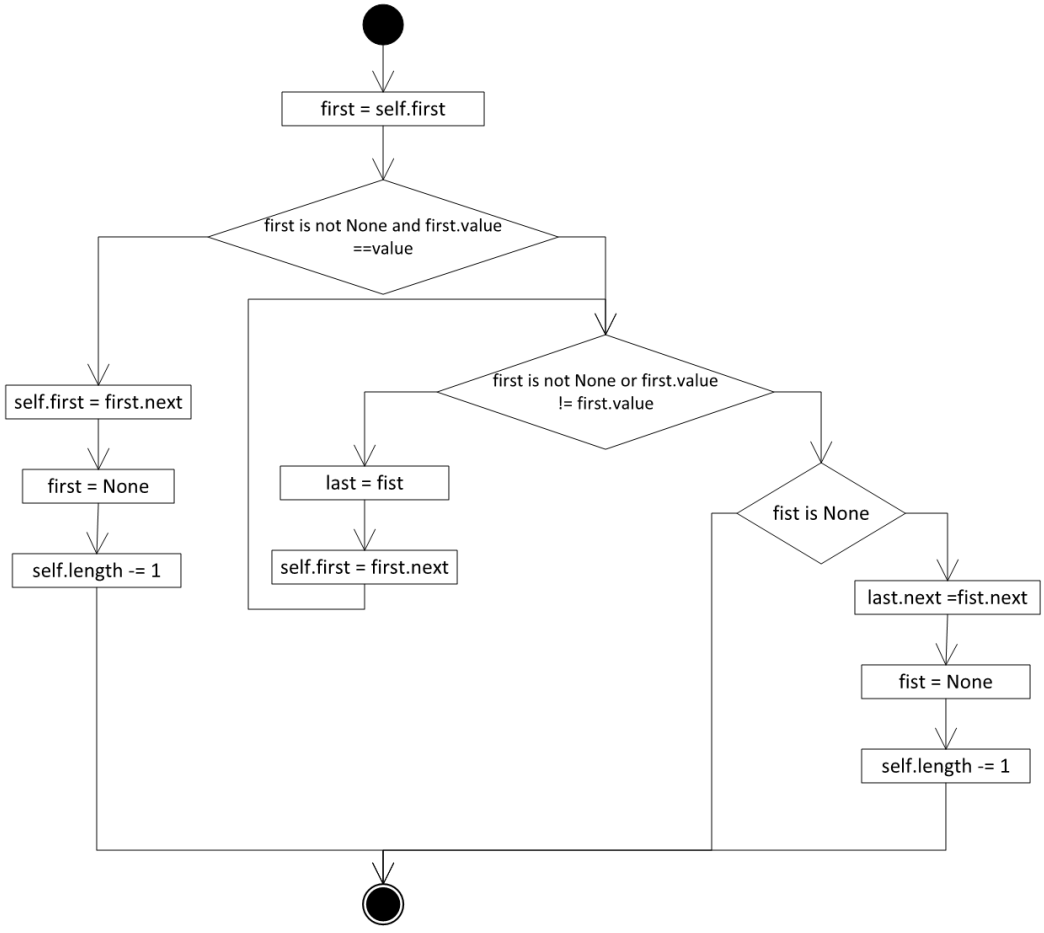

Рис. 11. Удаление элемента по его значению.

Поиск элемента по его значению представлен на Рис. 12. Диаграмма деятельности на Рис. 13.

```
def search(self, value):
    current = self.first
    count = 0
    while current is not None and current.value != value:
        count += 1
        current = current.next
    if current is None or current.value != value:
        count = -1
    return count
```

Рис. 12. Поиск элемента.

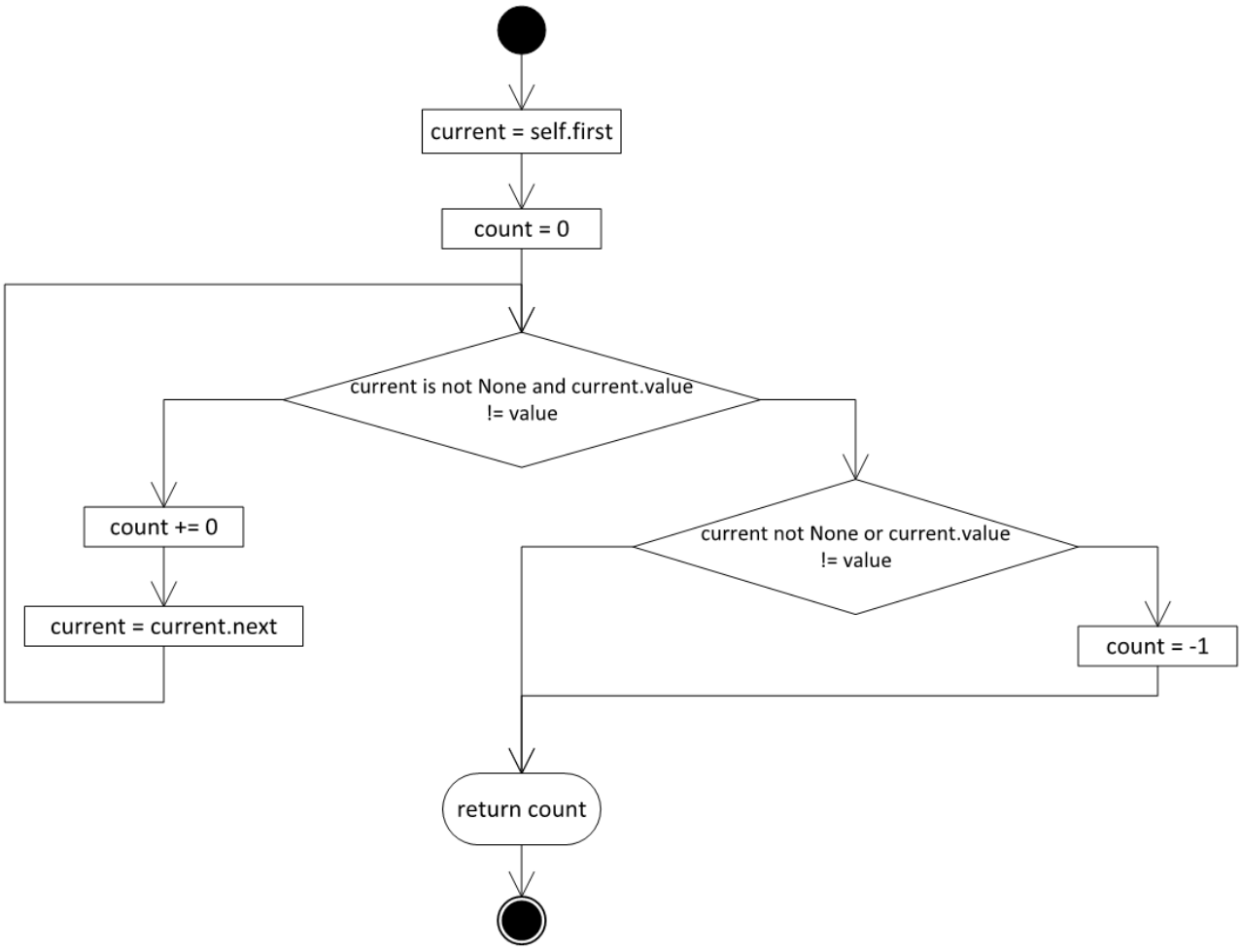

Рис. 13. Поиск элемента по его значению.

Код для решения задачи представлен на Рис. 14.

```python
def polynomial(a, x, n):
    if a >= n:
        P = Linked_List()
        for i in range(n, 0, -1):
            node = a * x ** i
            P.add(node)
            a -= 1
        return P
    else:
        return ValueError("a>=n")
```

Рис. 14. Код решения задачи.

Вывод: в ходе выполнения данной практической работы были изучены связанные списки и методы их реализации на языке Python.