Практическая работа №4

Тема: «Связанный список».

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

Ход работы:

Для реализации линейного списка сначала необходимо определить структуру узла.

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

Рис. 1 Структура узла

Метод добавления в начало списка представлен на рисунке 2

```
    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. 2 Метод добавления в начало списка

Диаграмма деятельности добавление элемента в начало списка приведена на рисунке 3.

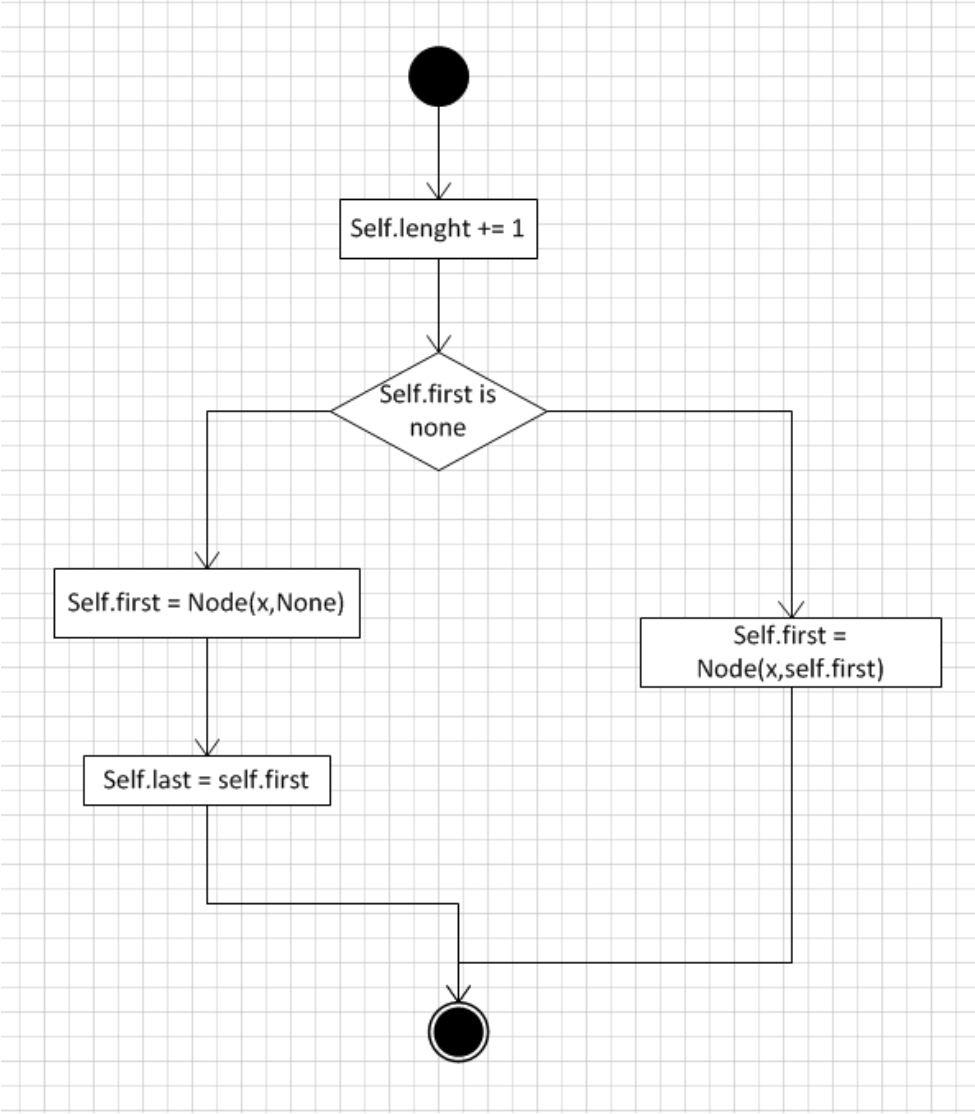

Рис. 3 Добавление элемента в начало списка.

Код метода добавления элемента в конец списка представлен на рисунке 4.

```
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

Диаграмма деятельности добавления элемента в конец списка приведена на рисунке 5.

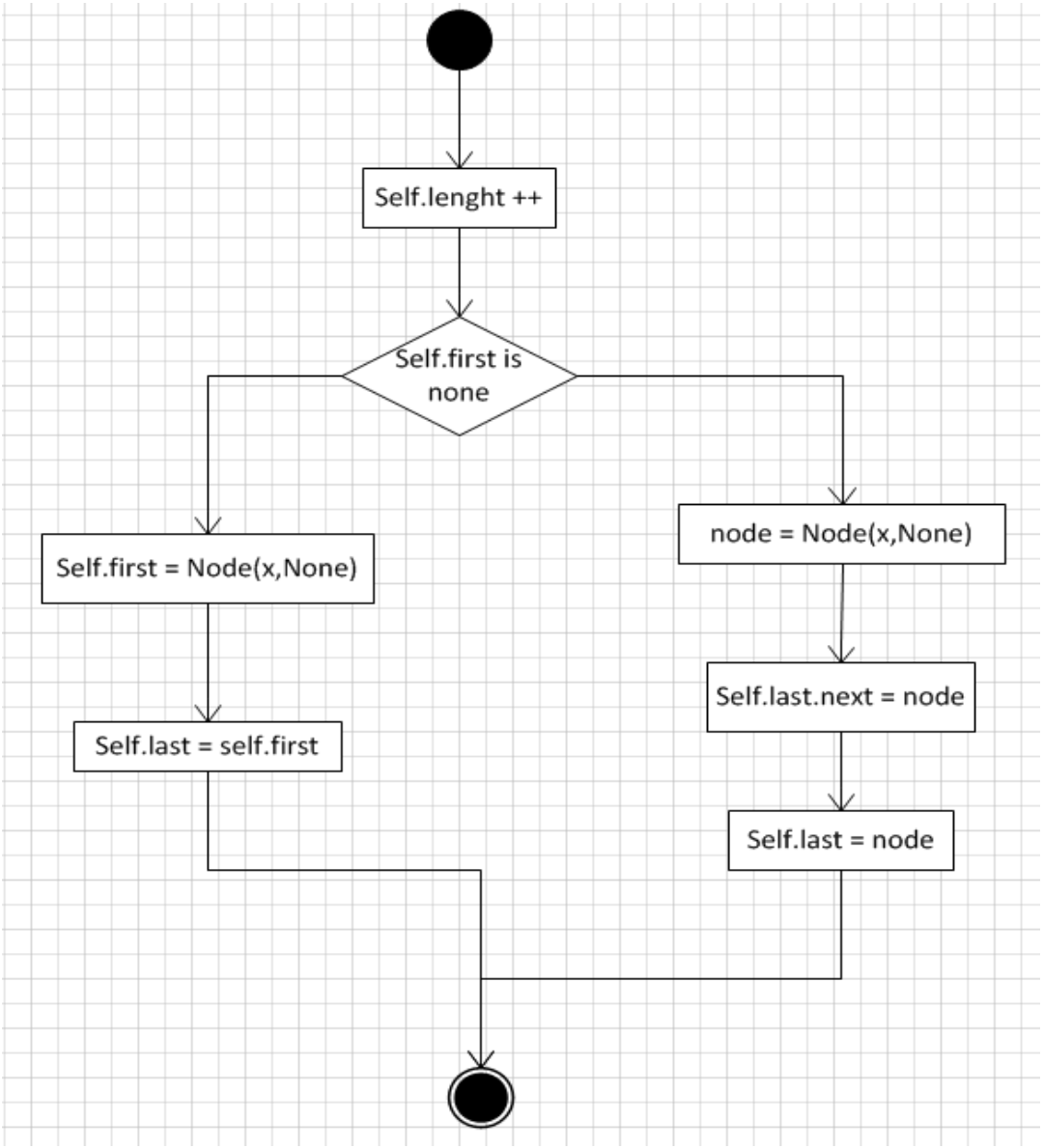

Рис. 5 Добавление элемента в конец списка

Метод вставки элемента в произвольное место приведён на рисунке 6.

```
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. 6 Код вставки элемента в произвольное место

Диаграмма деятельности для вставки элемента в произвольную позицию в списке приведена на рисунке 7.

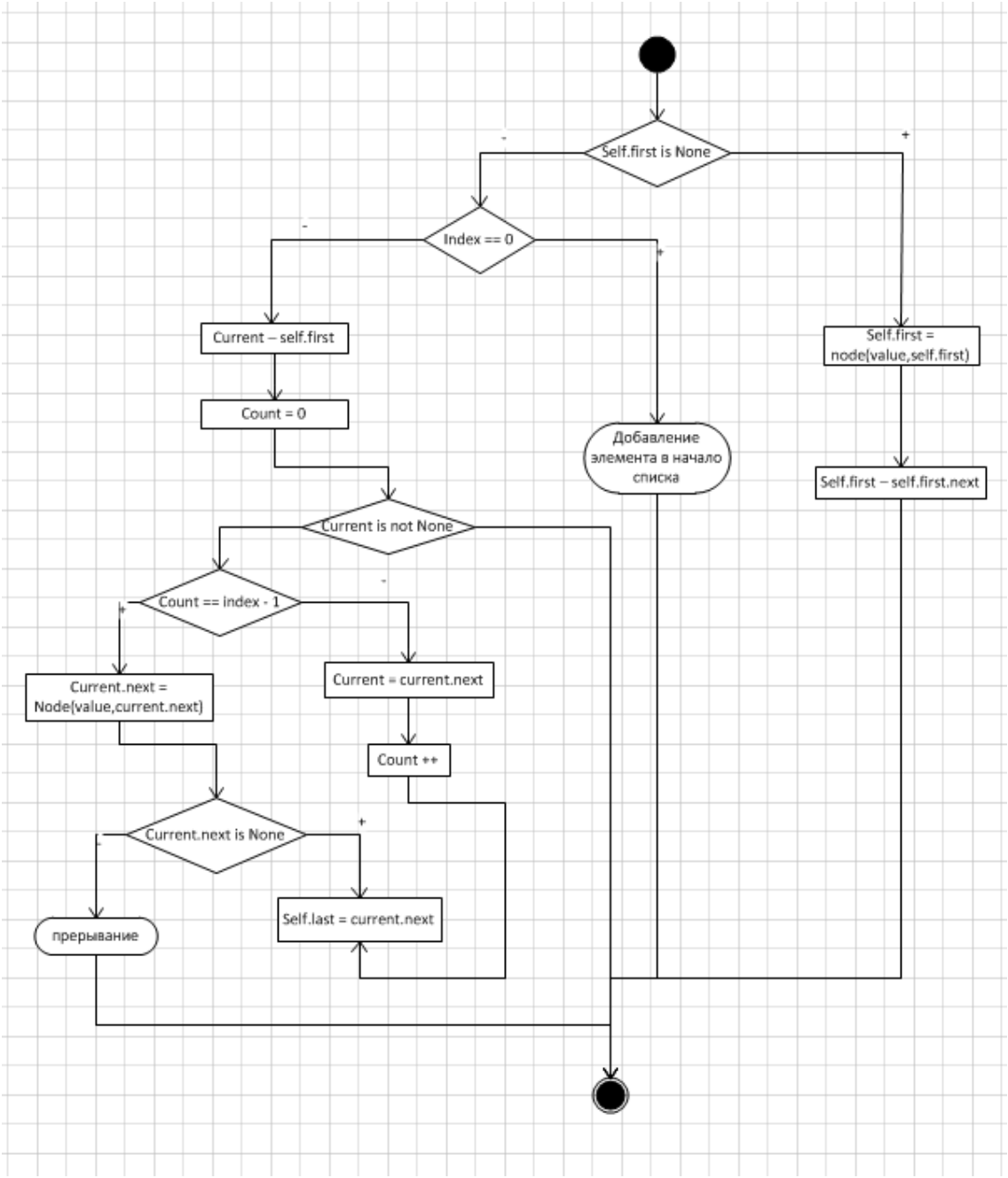

Рис. 7 Диаграмма деятельности для вставки элемента в произвольную позицию в списке.

Код для удаления головного элемента списка приведен на рисунке 8.

```
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

Диаграмма деятельности удаления головного элемента списка представлена на рисунке 9.

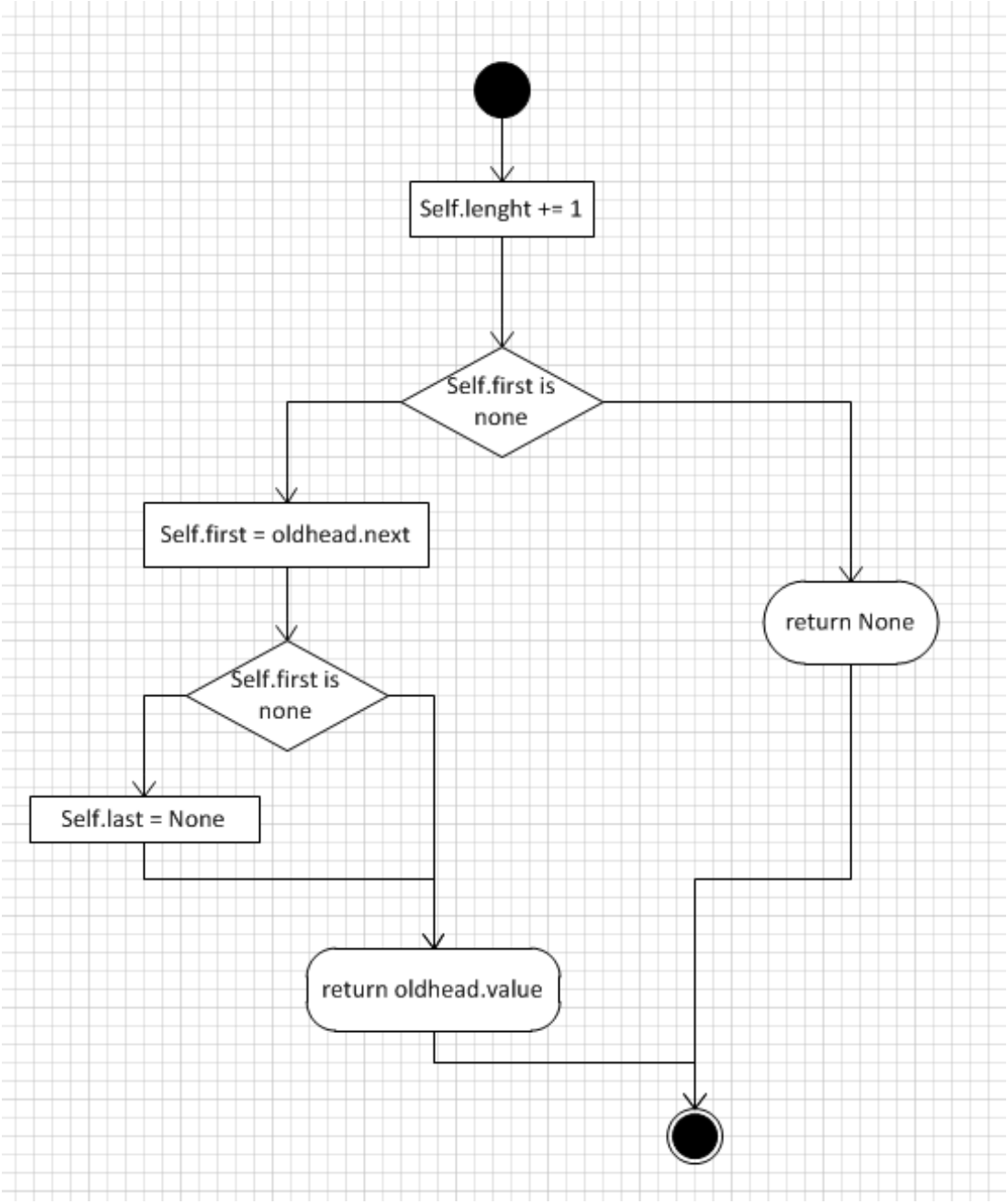

Рис. 9 Удаление головного элемента

Метод удаления элемента по его значению представлен на рисунке 10.

```
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

Диаграмма деятельности удаления элемента по его значению представлен на рисунок 11.

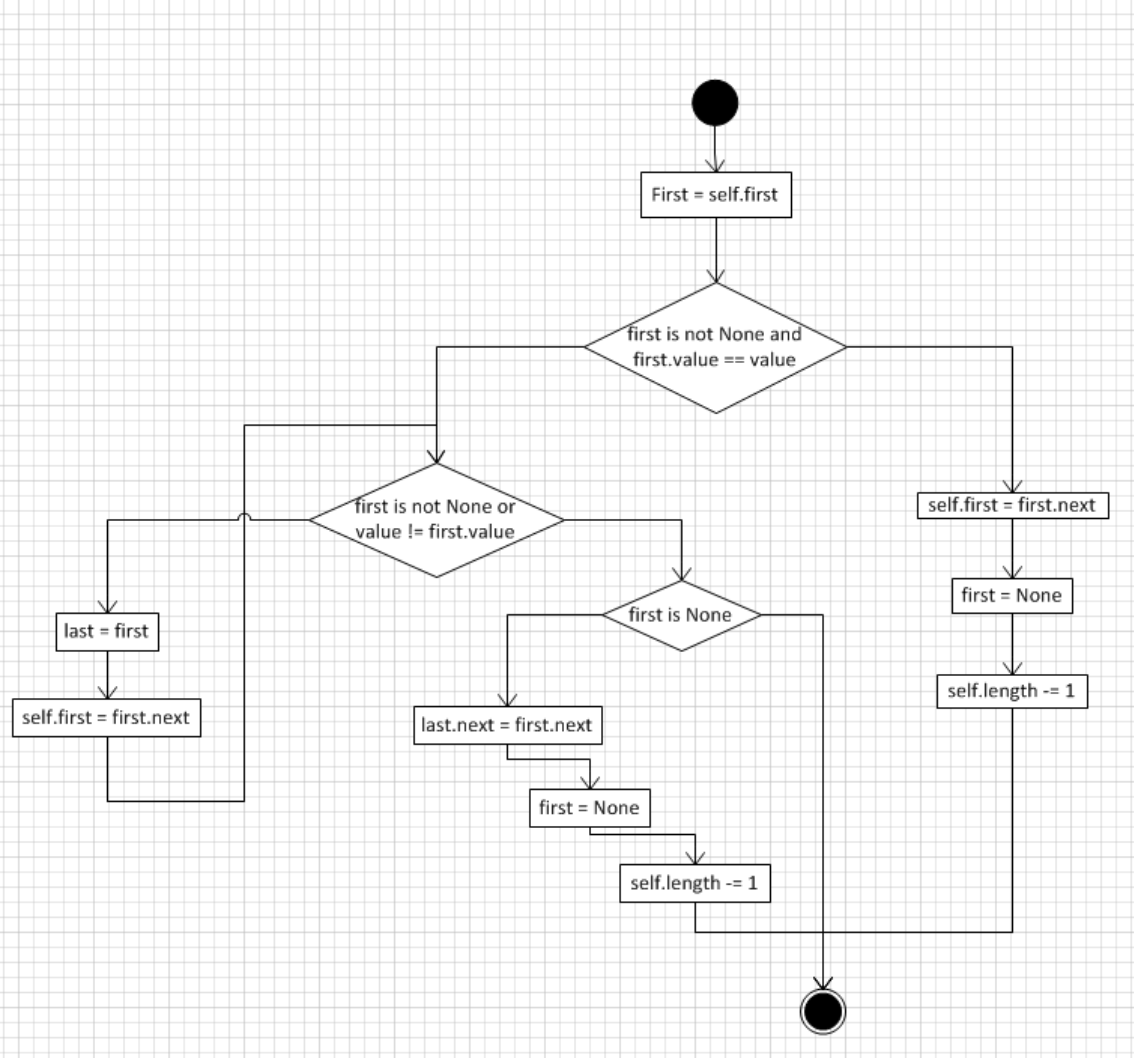

Рис. 11 Удаление элемента по его значению

Поиск элемента по его значению представлен на рисунке 12.

```
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

Диаграмма деятельности поиска элемента по его значению представлена на рисунке 13.

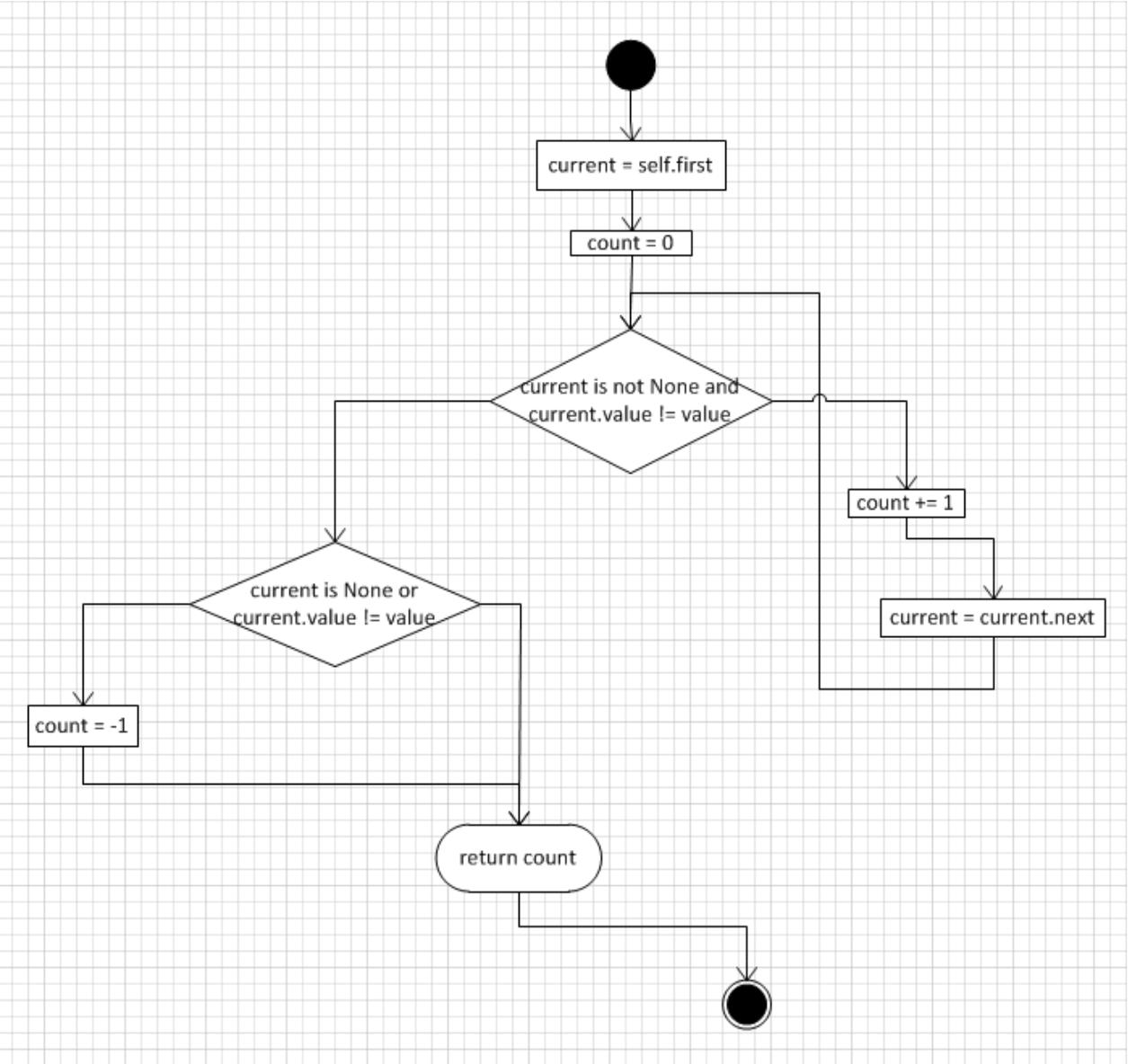

Рис. 13 Поиск элемента по его значению

Код для решения задачи представлен на рисунке 14.

```
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 данной практической работе были изучены связанные списки, и методы их программной реализации на высокоуровневом языке программирования Python.