Практическая работа №4

Тема: «Связанный список»

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

Для реализации «линейного списка» определим сначала структуру узла, код которого представлен в листинге 1.

Листинг 1. Структура узла.

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

Код метода вставки элемента в произвольное место представлен ниже в листинге .

Листинг . Код вставки элемента.

```
    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
```

Диаграмма деятельности для вставки элемента представлена на рисунке 1.

					АиСД.09.03.02.220000 ПР		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Третьяк И.Н.			Практическая работа №4 Связанный список	Лит.	Лист	Листов
Проверил		Береза А.Н.						
Реценз						ИСОиП (филиал) ДГТУ в г.Шахты ИСТ-Tb21		
Н. Контр.								
Утверд.								

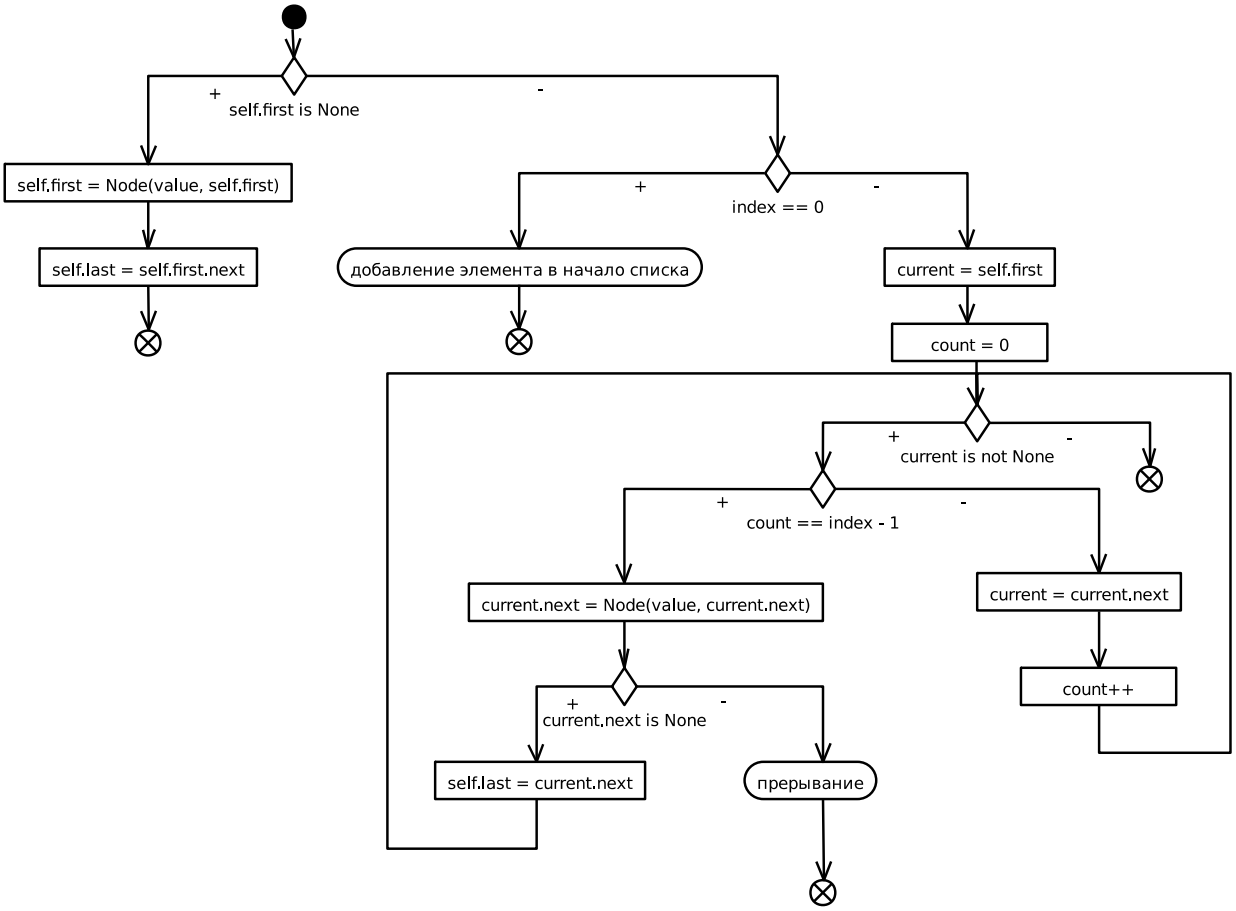

Рисунок 1 - Диаграмма деятельности для вставвки элемента в произвольную позицию в списке

Код метода добавления элемента в конец списка представлен в листинге 2. Диаграмма деятельности для этого метода приведена на рисунке 2.

Листинг 2. Реализация метода добавления элемента в конец списка.

```
    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
```

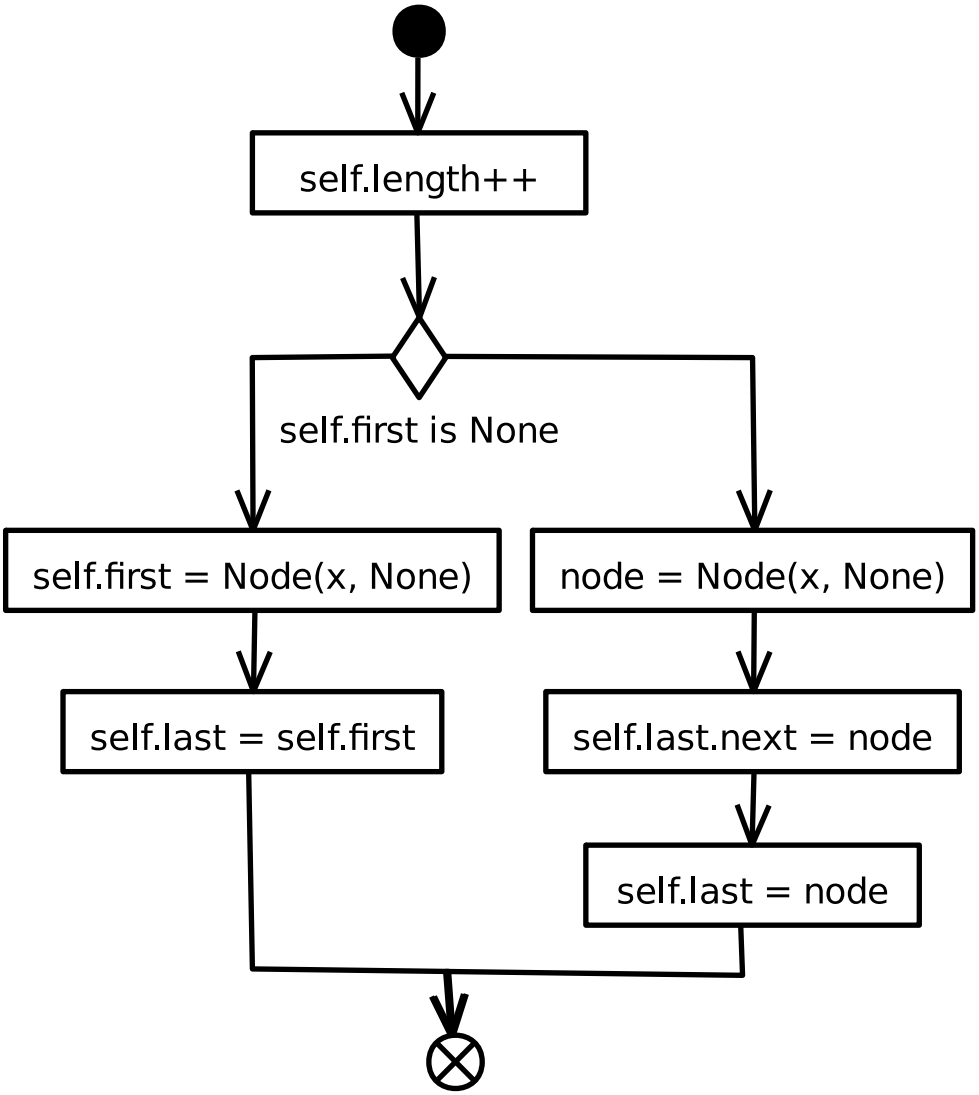

Рисунок 2 - Добавление элемента в конец списка

Метод добавления в начало списка представлен в листинге 3, а диаграмма деятельности для него приведена на рисунке 3.

Листинг 3. Метод добавления в начала списка.

```
    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)
```

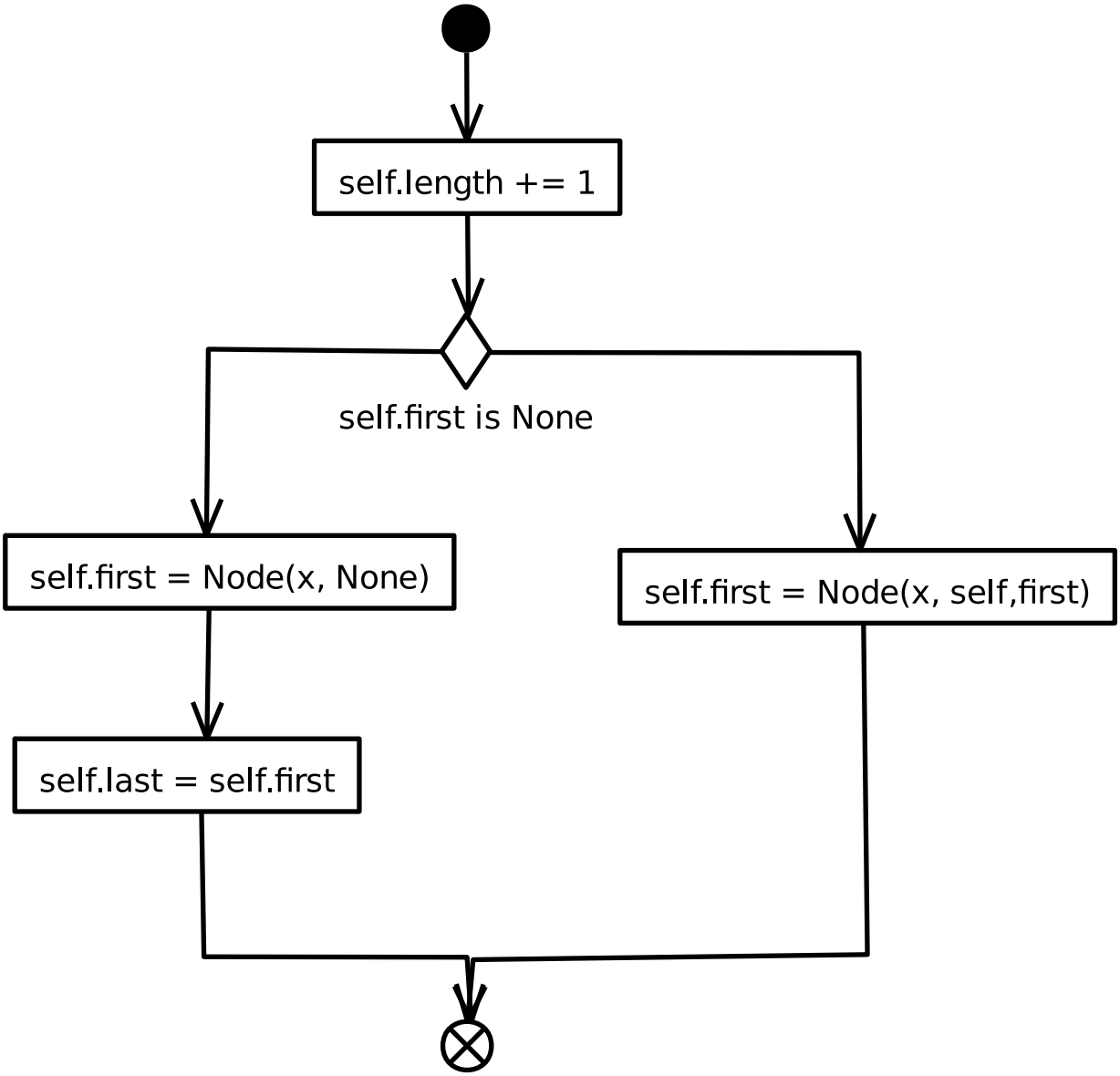

Рисунок 3 - Добавление элемента в начало списка

Код для удаления головного элемента списка приведен в листинге 4. Диаграмма деятельности для него представлена на рисунке 4.

Листинг 4. Удаление головного элемента.

```
    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
```

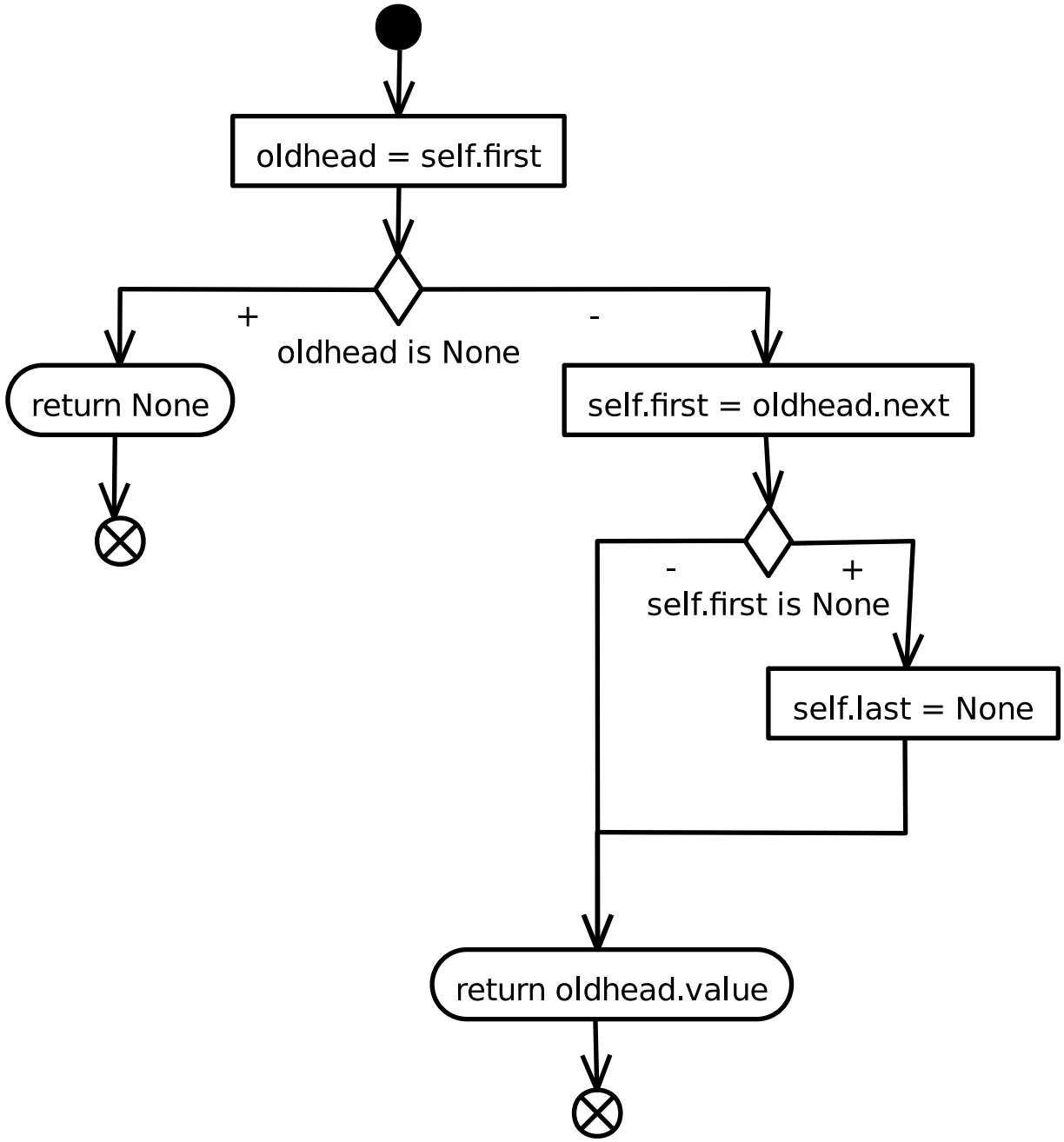

Рисунок 4 - Удаление головного элемента

Метод удаления элемента по его значению представлен в листинге 5. А диаграмма деятельности для него представлен на рисунке 5.

Листинг 5. Удаление элемента по его значению.

```
    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
```

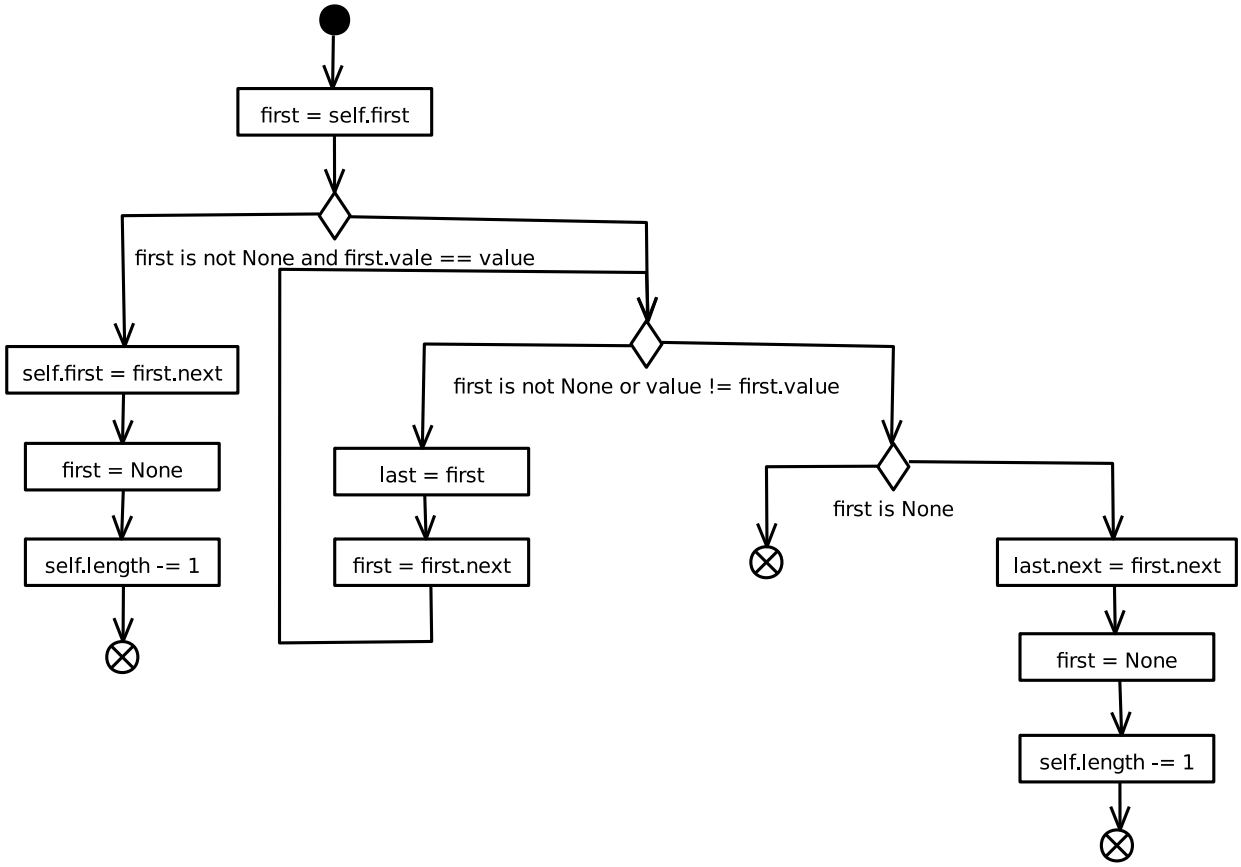

Рисунок 5 - Удаление элемента по его значению

Поиск элемента по его значению представлен в листинге 6. Диаграмма деятельности на рисунке 6.

Листинг 6. Поиск элемента.

```
    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
```

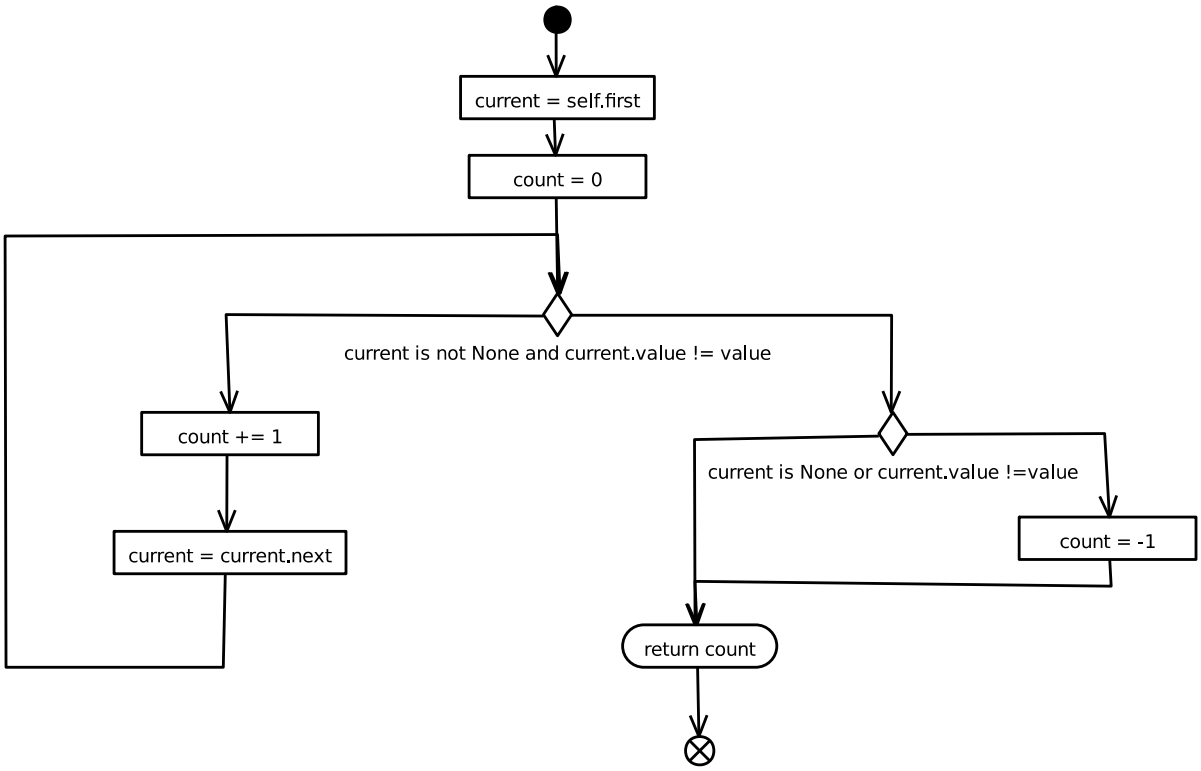

Рисунок 6 - Поиск элемента по его значению

Код для решения задачи представлен в листинге 7.

Листинг 7. Код решения задачи.

```
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")
```