Практическая работа №3

Тема: Хеш-таблицы

Цель работы: Изучить реализацию хеш-таблиц.

Хеш-таблица (hash table) — это специальная структура данных для хранения пар ключей и их значений. По сути это ассоциативный массив, в котором ключ представлен в виде хеш-функции. Где empty – флаг, указывающий, что ячейка свободна, в независимости от содержания там данных, а visit – флаг, указывающий, что ячейка просматривалась. Для реализации «Хеш-таблицы» опишем создание экземпляра класса и его заполнение:

```
class MyHash:
    hash_table: List[RecordHash]
    info: PInfo

    def __init__(self, size_table):
        self.size_table = size_table
        self.info = TInfo()
        self.hash_table = [RecordHash(info=self.info) for _ in r
ange(self.size_table)]
        self.size = 0
        self.step = 21
```

Опишем функция подсчета хеш-суммы значения:

```
def __hash_func(self, x):
        result = 0
        for i in range(len(x)):
            result += int(x[i]) * i
            result //= self.size_table
        return result
```

Диаграмма деятельности для функции подсчета хеш-суммы изображена на рисунке 1.

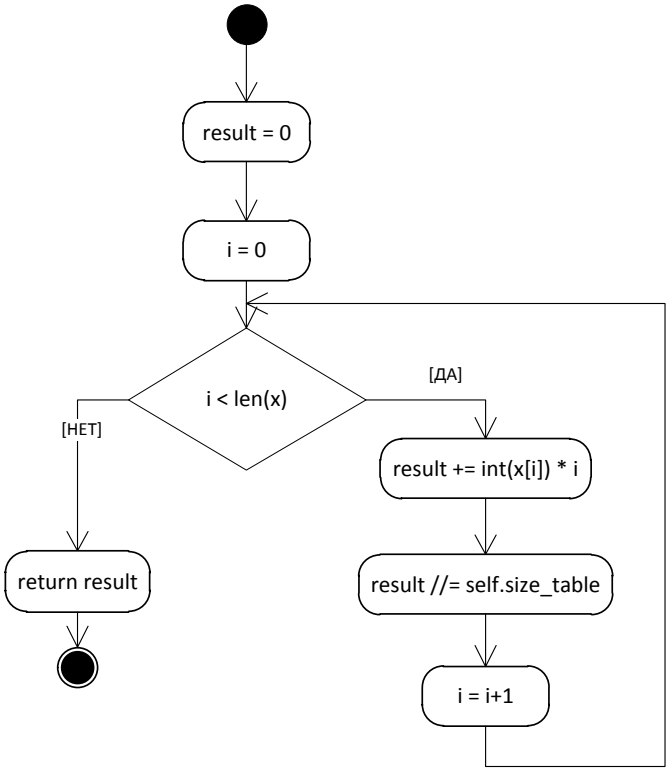

Рисунок 1 – Диаграмма деятельности для функции подсчета хеш-суммы.

Код метода добавления элемента в хеш-таблицу. Диаграмма деятельности метода добавления элемента в хеш-таблицу представлена на рисунке 2.

```
def add_hash(self, name: str, phone: str):
    pos = -1
    if self.size < self.size_table:
        pos = self.__hash_func(phone)
        while not self.hash_table[pos].empty:
            pos = (pos + self.step) // self.size_table
        self.hash_table[pos].empty = False
        self.hash_table[pos].visit = True
        contact = PInfo(phone=phone, name=name)
        self.hash_table[pos].info = contact
        self.size += 1
    return pos
```

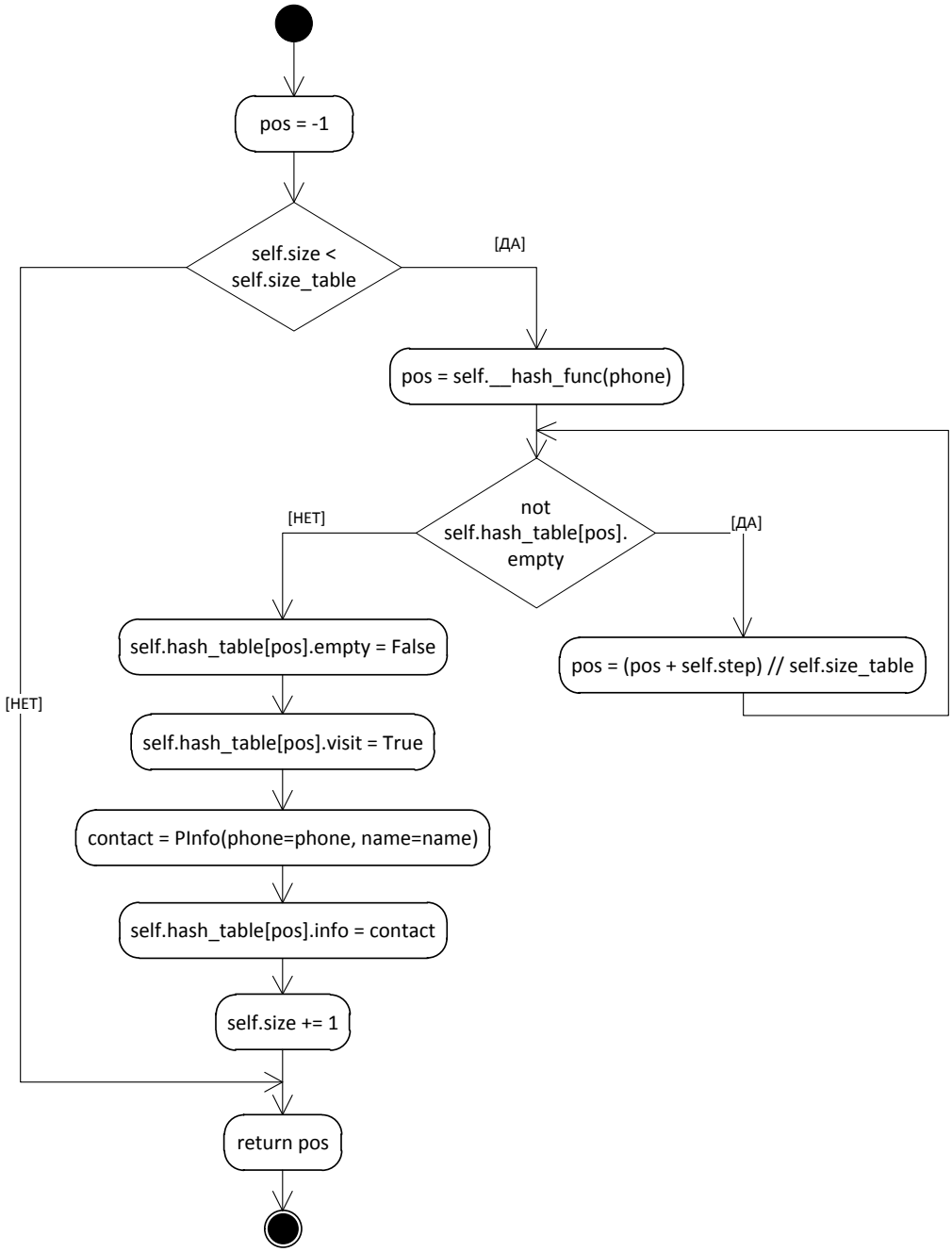

Рисунок 2 – Диаграмма деятельности метода добавления элемента в хеш-таблицу.

Для поиска элемента необходимо убедиться, что флаги visit каждой ячейки сброшены к дефолтным значениям, для этого мы используем функцию __clear_visit. Диаграмма деятельности для функции __clear_visit представлена на рисунке 3.

```
def __clear_visit(self):
    for i in self.hash_table:
        i.visit = False
```

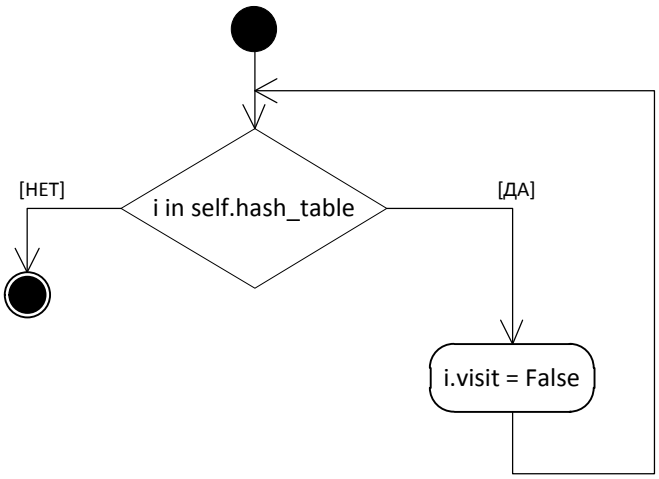

Рисунок 3 – Диаграмма деятельности для функции __clear_visit.

Код метода поиска значения в хеш-таблице. Диаграмма деятельности для поиска значения в хеш-таблице представлена на рисунке 4.

```
def search_hash(self, phone: str):
        result = -1
        ok: bool
        name = " "
        count = 1
        self.__clear_visit()
        i = self.__hash_func(phone)
        ok = self.hash_table[i].info.phone == phone
        while not ok and not self.hash_table[i].visit:
            count += 1
            self.hash_table[i].visit = True
            i = (i + self.step) // self.size_table
            ok = self.hash_table[i].info.phone == phone
        if ok:
            result = i
            name = self.hash_table[i].info.name
        return result
```

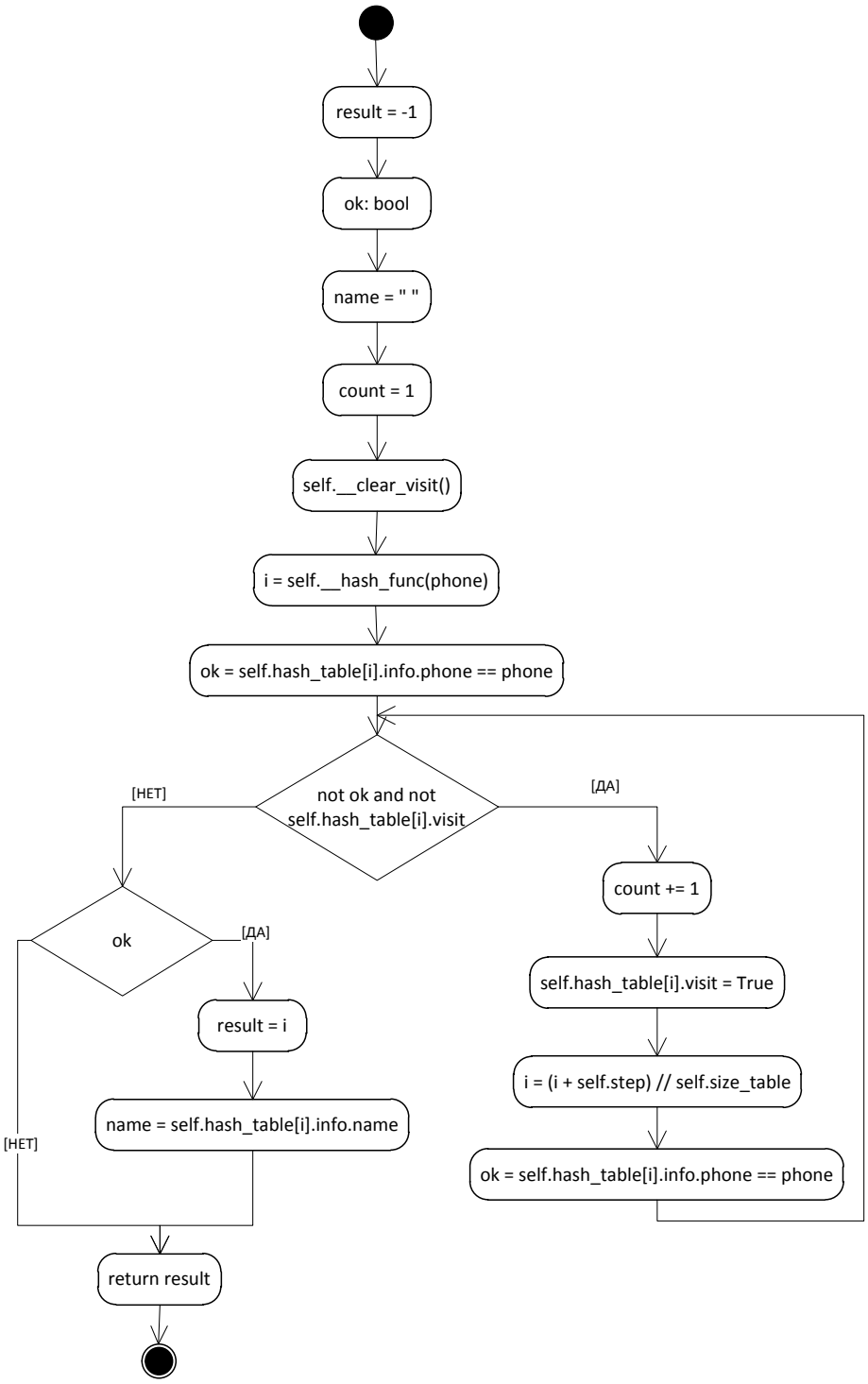

Рисунок 4 – Диаграмма деятельности метода поиска значения в хеш-таблице.

Код метода удаления элемента из хеш-таблицы. Необходимо найти нужный элемент и выставить флаг empty в позицию True. Диаграмма деятельности метода удаления элемента из хеш-таблицы представлена на рисунке 5.

```
def del_hash(self, phone: str):
        result = False
```

```
i = 0
if self.size != 0:
    i = self.__hash_func(phone)
    if self.hash_table[i].info.phone == phone:
        self.hash_table[i].empty = True
        result = True
        self.size -= 1
    else:
        i = self.search_hash(phone)
        if (i != -1):
            self.hash_table[i].empty = True
            result = True
            self.size -= 1
return result
```

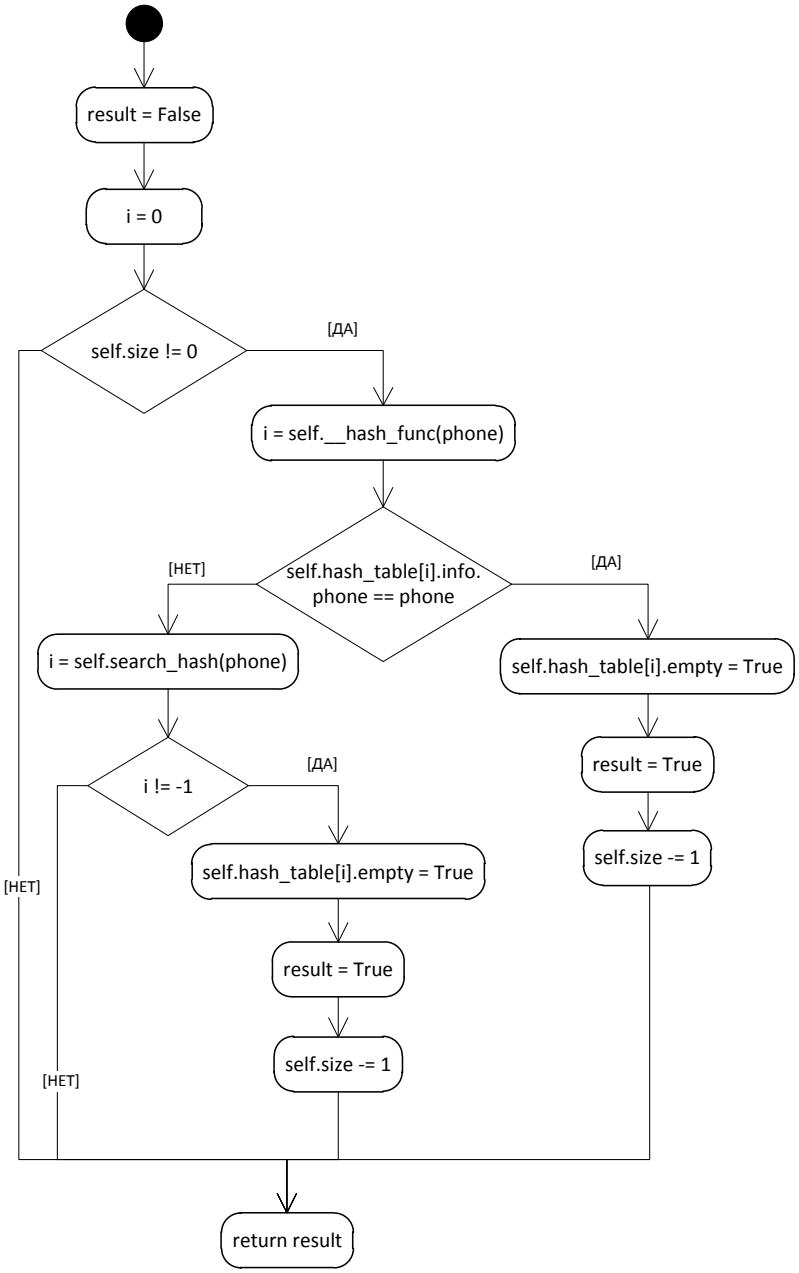

Рисунок 5 – Диаграмма деятельности метода удаления элемента из хеш-таблицы.

Реализация телефонного справочника:

```
    def __str__(self):
        out = ""
        head = "{:<5}{:<25}{:<25}".format("N", "NAME", "PHONE")
        out += head
        out += "\n"
        for i in range(self.size_table):
            name: str = self.hash_table[i].info.name
            phone: str = self.hash_table[i].info.phone
            string = "{:<5}{:<25}{:<25}".format(i + 1, name, pho
ne)
            out += string
            out += "\n"
        return out

table = MyHash(5)
table.add_hash("Dmitry Klejmenkin", "89000000000")
print(table)
```

Скриншот рабочей программы представлен на рисунке 6:

```
N    NAME                      PHONE
1    Dmitry Klejmenkin         89000000000
2
3
4
5
```

Рисунок 6 – Скриншот рабочей программы.

Вывод. В ходе выполнения практической работы мы ознакомились с хеш-таблицами, а так-же методами их реализации.