САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ

ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ

ФАКУЛЬТЕТ ИНФОКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ

Отчет по лабораторной работе №6

по курсу «Алгоритмы и структуры данных»

Тема: Тема работы

Вариант 23

Выполнил:

Юшков А.М.

К3139

Санкт-Петербург

2025г.

# Содержание отчета

[**Содержание отчета**](#_10udeakjagvs) **2**

[**Задачи по варианту**](#_gf7yxvsg0zb) **3**

[Задача №1.](#_pgzbaj56cf0o) Множества 3

[Задача №2.](#_pgzbaj56cf0o) Телефонная книга 4

[Задача №4.](#_pgzbaj56cf0o) Прошитый ассоциативный массив 6

[**Вывод**](#_fu90fuyk873) **8**

# Задачи по варианту

## Задача №1. Множества

from lab6.utils import read, write  
  
  
def operations(data):  
 result = []  
 current\_set = set()  
 for line in data:  
 operation, x = line  
 if operation == 'A':  
 current\_set.add(x)  
 elif operation == 'D':  
 current\_set.discard(x)  
 elif operation == '?':  
 if x in current\_set:  
 result.append('Y')  
 else:  
 result.append('N')  
 return result  
  
  
def main():  
 write(end='')  
 array = read(type\_convert=str)  
 result = operations(array)  
 for line in result:  
 write(line, to\_end=True)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Текстовое объяснение решения.

Программа эффективно обрабатывает операции над множеством, используя встроенную структуру данных set. Это позволяет быстро выполнять добавление, удаление и проверку принадлежности элемента. Результаты запросов выводятся в порядке их поступления

Описание алгоритма

**Входные данные**:

Программа получает список операций, где каждая операция представлена в виде кортежа:

('A', x) — добавить элемент x в множество.

('D', x) — удалить элемент x из множества.

('?', x) — проверить, принадлежит ли элемент x множеству.

**Обработка операций**:

Используется структура данных set для хранения элементов.

Для каждой операции:

Если операция 'A', элемент добавляется в множество.

Если операция 'D', элемент удаляется из множества (если он существует).

Если операция '?', программа проверяет, принадлежит ли элемент множеству, и сохраняет результат ('Y' или 'N').

**Результат**:

Программа возвращает список результатов для всех операций '?'.

Результат работы кода на примере из текста задачи

Input.txt

A 2  
A 5  
A 3  
? 2  
? 4  
A 2  
D 2  
? 2

Output.txt

Y  
N  
N

## Задача №2. Телефонная книга

from lab6.utils import \*  
  
def phone\_book(data):  
 book = {}  
 results = []  
  
 for command in data:  
 if command[0] == "add":  
 number, name = command[1], command[2]  
 book[number] = name  
 elif command[0] == "del":  
 number = command[1]  
 book.pop(number, None)  
 elif command[0] == "find":  
 number = command[1]  
 results.append(book.get(number, "not found"))  
 return results  
  
  
def main():  
 write(end='')  
 array = read(type\_convert=str)  
 result = phone\_book(array)  
 for line in result:  
 write(line, to\_end=True)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Программа эффективно управляет телефонной книгой, используя словарь для хранения и быстрого доступа к данным. Это позволяет выполнять операции добавления, удаления и поиска за время O(1).

Алгоритм:

Входные данные:

Входные данные:

Программа получает список команд, где каждая команда представлена в виде кортежа:

("add", number, name) — добавить контакт с номером number и именем name.

("del", number) — удалить контакт с номером number.

("find", number) — найти имя контакта по номеру number.

Обработка команд:

Используется словарь book для хранения контактов, где ключом является номер телефона, а значением — имя.

Для каждой команды:

Если команда "add", номер и имя добавляются в словарь.

Если команда "del", номер удаляется из словаря (если он существует).

Если команда "find", программа ищет имя по номеру и сохраняет результат (name или "not found").

Результат:

Программа возвращает список результатов для всех команд "find".

Результат работы кода на примере из текста задачи

Input.txt

add 911 police  
add 76213 Mom  
add 17239 Bob  
find 76213  
find 910  
find 911  
del 910  
del 911  
find 911  
find 76213  
add 76213 daddy  
find 76213

Output.txt

Mom  
not found  
police  
not found  
Mom  
daddy

## Задача №4. Прошитый ассоциативный массив

from lab6.utils import \*  
from collections import OrderedDict  
  
  
def associative\_array(commands):  
 assoc\_array = OrderedDict()  
 results = []  
  
 for parts in commands:  
 command = parts[0]  
  
 if command == "put":  
 x, y = parts[1], parts[2]  
 if x in assoc\_array:  
 assoc\_array[x] = y  
 else:  
 assoc\_array[x] = y  
  
 elif command == "get":  
 x = parts[1]  
 results.append(assoc\_array.get(x, "<none>"))  
  
 elif command == "prev":  
 x = parts[1]  
 if x in assoc\_array:  
 keys = list(assoc\_array.keys())  
 idx = keys.index(x)  
 if idx > 0:  
 results.append(assoc\_array[keys[idx - 1]])  
 else:  
 results.append("<none>")  
 else:  
 results.append("<none>")  
  
 elif command == "next":  
 x = parts[1]  
 if x in assoc\_array:  
 keys = list(assoc\_array.keys())  
 idx = keys.index(x)  
 if idx < len(keys) - 1:  
 results.append(assoc\_array[keys[idx + 1]])  
 else:  
 results.append("<none>")  
 else:  
 results.append("<none>")  
  
 elif command == "delete":  
 x = parts[1]  
 assoc\_array.pop(x, None)  
  
 return results  
  
  
def main():  
 write(end='')  
 array = read(type\_convert=str)  
 result = associative\_array(array)  
 for line in result:  
 write(line, to\_end=True)  
  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

Программа эффективно управляет ассоциативным массивом, поддерживая не только стандартные операции (добавление, удаление, поиск), но и расширенные функции, такие как получение предыдущего и следующего значения по ключу. Использование OrderedDict позволяет сохранять порядок добавления элементов

Алгоритм:

**Входные данные**:

Программа получает список команд, где каждая команда представлена в виде кортежа:

* + - ("put", x, y) — добавить или обновить значение y для ключа x.
    - ("get", x) — получить значение для ключа x (или "<none>", если ключ отсутствует).
    - ("prev", x) — получить значение для ключа, который был добавлен перед ключом x (или "<none>", если такого ключа нет).
    - ("next", x) — получить значение для ключа, который был добавлен после ключа x (или "<none>", если такого ключа нет).
    - ("delete", x) — удалить ключ x из массива.

**Обработка команд**:

Используется структура данных OrderedDict (упорядоченный словарь) для хранения пар ключ-значение с сохранением порядка добавления.

Для каждой команды:

* + - "put": Добавляет или обновляет значение для ключа.
    - "get": Возвращает значение по ключу или "<none>", если ключ отсутствует.
    - "prev": Возвращает значение для предыдущего ключа в порядке добавления.
    - "next": Возвращает значение для следующего ключа в порядке добавления.
    - "delete": Удаляет ключ из массива.

**Результат:**

Программа возвращает список результатов для команд "get", "prev" и "next".

Результат работы кода на примере из текста задачи

Input.txt

put zero a  
put one b  
put two c  
put three d  
put four e  
get two  
prev two  
next two  
delete one  
delete three  
get two  
prev two  
next two  
next four

Output.txt

c  
b  
d  
c  
a  
e  
<none>

# Вывод

В ходе выполнения лабораторной работы были разработаны и протестированы три программы, каждая из которых решает уникальную задачу, связанную с обработкой данных и использованием различных структур данных. Все программы демонстрируют эффективное применение ключевых концепций программирования, таких как работа с множествами, словарями и упорядоченными структурами данных.