ГУАП

КАФЕДРА № 42

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент |  |  |  | С. А. Чернышев |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ № 7 |
| ТЕЛЕФОННАЯ КНИГА |
| по курсу: |
| АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ гр. № | 4326 |  |  |  | Г. С. Томчук |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

СОДЕРЖАНИЕ

[1 Цель работы 3](#_Toc185673883)

[2 Задание 3](#_Toc185673884)

[3 Краткое описание хода разработки 3](#_Toc185673885)

[4 Исходный код программы 4](#_Toc185673886)

[5 Реализация основного и двух любых дополнительных заданий из списка 6](#_Toc185673887)

[6 Результаты работы программы с примерами 7](#_Toc185673888)

[7 Выводы 8](#_Toc185673889)

1. Цель работы

Целью данной лабораторной работы является создать структуру данных для телефонной книги, поддерживающую операции добавления, удаления и поиска контактов по номеру. Реализовать основные операции с учетом ограничений на номера и имена, а также два дополнительных задания: поиск по частичному совпадению номера и поддержка групп и меток для контактов.

1. Задание

Разработать программу, которая:

1. Обрабатывает команды:

* add number name — добавляет контакт. Если номер уже существует, обновляет имя.
* del number — удаляет контакт по номеру.
* find number — находит имя по номеру или выводит «not found».

1. Реализует два дополнительных задания:

* Поиск по частичному совпадению номера.
* Поддержку групп и меток для контактов.

Программа должна быть оптимизирована для работы с большими данными.

1. Краткое описание хода разработки
   * + 1. Изучены ограничения задачи, такие как длина номера, допустимые символы в имени, и количество запросов. Определено, что для хранения данных подойдет словарь (хеш-таблица), поскольку он обеспечивает доступ к данным за O(1) в среднем случае.
       2. Для основной задачи создан класс PhoneBook, содержащий словарь для хранения пар "номер-имя". Дополнительно реализованы структуры для хранения меток и групп.
       3. Основной функционал (добавление, удаление, поиск по номеру) реализован через методы класса:
       - add\_contact: Добавляет новый контакт или обновляет существующий.
       - delete\_contact: Удаляет контакт, если он существует.
       - find\_contact: Возвращает имя или «not found», если номер отсутствует.
       1. Реализованы два дополнительных задания:
       - Поиск по частичному совпадению номера: Используется метод find\_partial, который находит все номера, начинающиеся с заданной строки.
       - Поддержка групп и меток: Контакты можно добавлять в группы и назначать им метки. Реализованы методы для поиска по группе или метке.
       1. Запросы собираются в список, затем обрабатываются пакетно. Это позволяет поддерживать требуемый формат вывода, где результаты поиска выводятся после обработки всех запросов.
       2. Программа протестирована на данных с разным количеством запросов, чтобы убедиться в корректной работе и соблюдении временных ограничений.
2. Исходный код программы

class PhoneBook:

def \_\_init\_\_(self):

self.contacts = {} *# Основной справочник {номер: {name, group, tags}}*

self.groups = {} *# Словарь групп {group\_name: [numbers]}*

self.tags = {} *# Словарь меток {tag\_name: [numbers]}*

def add\_contact(self, number, name, group=None):

if number in self.contacts:

self.contacts[number]['name'] = name

else:

self.contacts[number] = {'name': name, 'group': group, 'tags': set()}

if group:

if group not in self.groups:

self.groups[group] = []

if number not in self.groups[group]:

self.groups[group].append(number)

def delete\_contact(self, number):

if number in self.contacts:

group = self.contacts[number]['group']

if group and number in self.groups.get(group, []):

self.groups[group].remove(number)

for tag in self.contacts[number]['tags']:

if number in self.tags[tag]:

self.tags[tag].remove(number)

del self.contacts[number]

def find\_contact(self, number):

return self.contacts.get(number, {}).get('name', 'not found')

def add\_tag(self, number, tag\_name):

if number in self.contacts:

self.contacts[number]['tags'].add(tag\_name)

if tag\_name not in self.tags:

self.tags[tag\_name] = []

if number not in self.tags[tag\_name]:

self.tags[tag\_name].append(number)

def find\_by\_group(self, group\_name):

if group\_name in self.groups:

return [self.contacts[number]['name'] for number in self.groups[group\_name]]

return []

def find\_by\_tag(self, tag\_name):

if tag\_name in self.tags:

return [self.contacts[number]['name'] for number in self.tags[tag\_name]]

return []

def find\_partial(self, partial\_number):

result = []

for number in self.contacts:

if str(number).startswith(partial\_number):

result.append((number, self.contacts[number]['name']))

return result

def process\_requests(requests):

phone\_book = PhoneBook()

results = []

for request in requests:

query = request.split()

if query[0] == "add":

number, name = query[1], query[2]

group = query[3] if len(query) > 3 else None

phone\_book.add\_contact(number, name, group)

elif query[0] == "del":

number = query[1]

phone\_book.delete\_contact(number)

elif query[0] == "find":

if len(query) == 2:

number = query[1]

results.append(phone\_book.find\_contact(number))

elif query[1] == "partial":

partial\_number = query[2]

matches = phone\_book.find\_partial(partial\_number)

if matches:

results.extend([f"{number}: {name}" for number, name in matches])

else:

results.append("not found")

elif query[0] == "tag":

number, tag\_name = query[1], query[2]

phone\_book.add\_tag(number, tag\_name)

elif query[0] == "find\_by\_group":

group\_name = query[1]

names = phone\_book.find\_by\_group(group\_name)

results.append(", ".join(names) if names else "not found")

elif query[0] == "find\_by\_tag":

tag\_name = query[1]

names = phone\_book.find\_by\_tag(tag\_name)

results.append(", ".join(names) if names else "not found")

return results

def main():

q = int(input())

requests = []

for \_ in range(q):

requests.append(input())

results = process\_requests(requests)

print("\n".join(results))

if \_\_name\_\_ == "\_\_main\_\_":

main()

1. Реализация основного и двух любых дополнительных заданий из списка

Дополнительное задание 1: Поиск по частичному совпадению номера. Метод find\_partial итерируется по ключам словаря, чтобы найти номера, начинающиеся с заданной подстроки.

Дополнительное задание 2: Поддержка групп и меток. Для реализации групп и меток добавлены дополнительные словари. Методы add\_to\_group и add\_tag назначают контактам группы и метки, а find\_by\_group и find\_by\_tag позволяют искать контакты по этим критериям.

1. Результаты работы программы с примерами

На рис. 1, 2, 3 изображено тестирование написанной программы с различными входными данными.

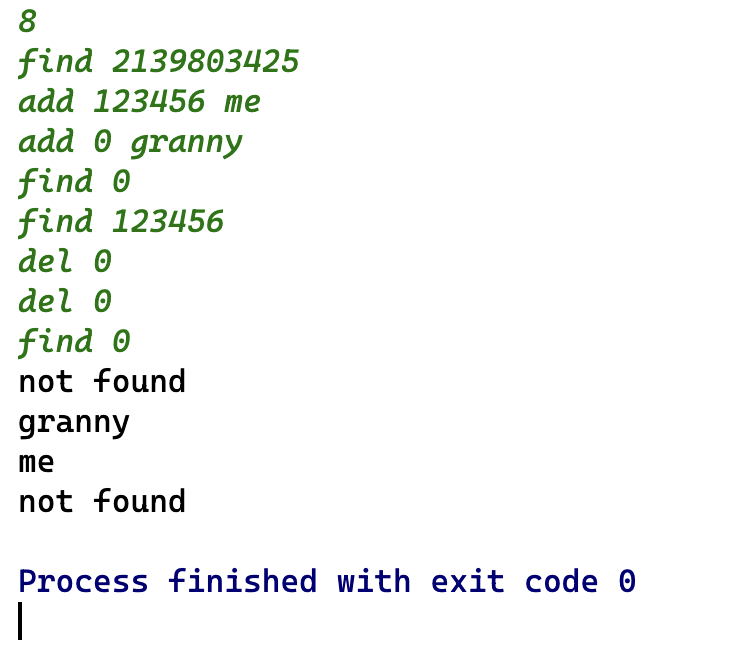


Рисунок 1

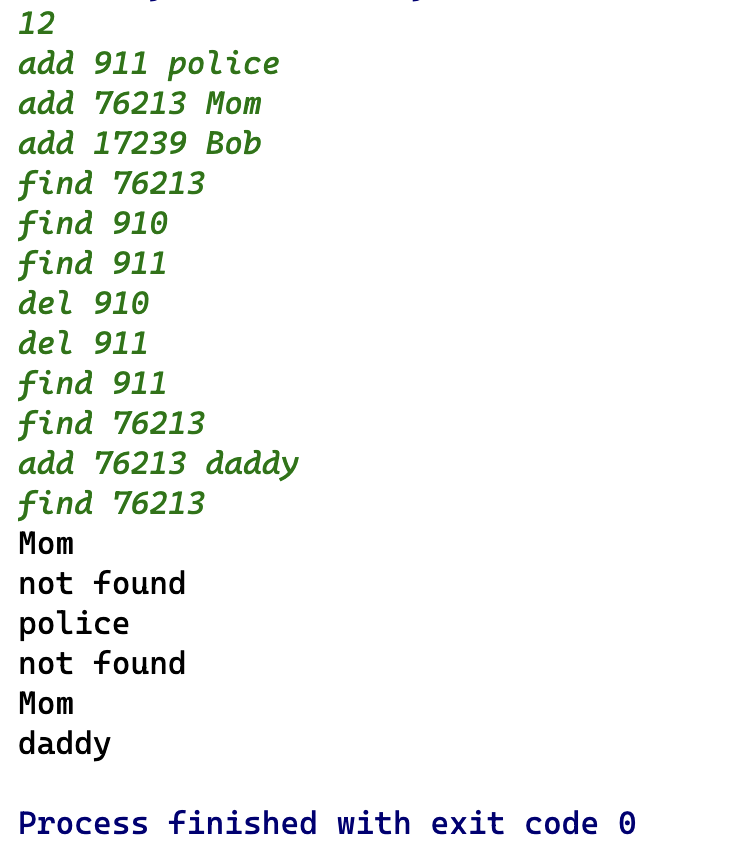


Рисунок 2

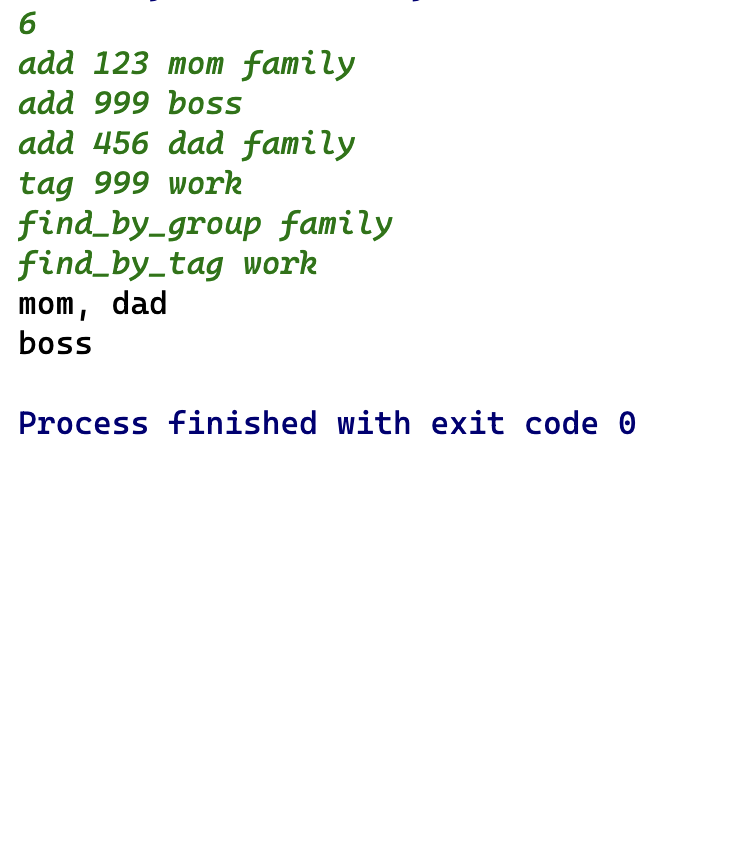


Рисунок 3

1. Выводы
   * + - Создана эффективная структура данных для управления телефонной книгой, соответствующая всем базовым требованиям задачи.

* Поиск по частичному совпадению номера облегчает работу с большим количеством контактов.
* Группы и метки позволяют структурировать телефонную книгу, добавляя гибкость в ее использование.
* Использование словарей обеспечивает высокую производительность операций добавления, удаления и поиска.
* Навыки, полученные в процессе работы:
  + Проектирование структур данных.
  + Реализация алгоритмов для сложных запросов.
  + Оптимизация работы программы с учетом ограничений.