МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА № 42

ОТЧЁТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

| старший преподаватель |  |  |  | С.Ю. Гуков |
| --- | --- | --- | --- | --- |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

| ОТЧЁТ О ЛАБОРАТОРНОЙ РАБОТЕ №7 |
| --- |
| Телефонная книга |
| по курсу: |
| АЛГОРИТМЫ И СТРУКТУРЫ ДАННЫХ |
|  |

РАБОТУ ВЫПОЛНИЛ

| СТУДЕНТ гр. № | 4329 |  |  |  | Д.С. Шаповалова |
| --- | --- | --- | --- | --- | --- |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2024

ОГЛАВЛЕНИЕ

[Цель работы 3](#_30j0zll)

[Постановка задачи 3](#_1fob9te)

[Схема алгоритма решения 4](#_3znysh7)

[Полное описание реализованной функции 4](#_gqc1nhfmq022)

[Листинг программы 5](#_2et92p0)

[Результат выполнения программы. 5](#_tyjcwt)

[ВЫВОДЫ 6](#_3dy6vkm)

# Цель работы

Реализовать структуру данных, эффективно обрабатывающую запросы вида add number name, del number и find number.

# Постановка задачи

Задание: Написать программу, обрабатывающую последовательность запросов добавить/найти/удалить номер в “телефонную книгу”. Текст задания приведён в таблице 1.

Таблица 1. Индивидуальное задание

| № | Текст задания | Вход | Выход |
| --- | --- | --- | --- |
| 0 | Реализовать простую телефонную книгу, поддерживающую три следующих типа запросов. С указанными ограничениями данная задача может быть решена с использованием таблицы с прямой адресацией.  *add number name*: добавить запись с именем name и телефонным номером number. Если запись с таким телефонным номером уже есть, нужно заменить в ней имя на name.  *del number*: удалить запись с соответствующим телефонным номером. Если такой записи нет, ничего не делать.  *find number*: найти имя записи с телефонным номером number. Если запись с таким номером есть, вывести имя. В противном случае вывести «not found» (без кавычек).  **Ограничения**. 1 ≤ n ≤ 105. Телефонные номера содержат не более семи цифр и не содержат ведущих нулей. Имена содержат только буквы латинского алфавита, не являются пустыми строками и имеют длину не больше 15. Гарантируется, что среди имён не встречается строка «not found».  Доп. усложнения:  (4) Защита от дубликатов имен. Ввести правило, что одно имя может быть связано только с одним номером. Если пользователь пытается добавить имя, которое уже существует с другим номером, программа должна выдать предупреждение или проигнорировать операцию.  (7) Лимит по количеству контактов. Ввести ограничение на максимальное количество контактов в телефонной книге. При достижении лимита программа должна выдавать ошибку на добавление новых контактов. | Первая строка содержит число запросов n. Каждая из следующих n строк задаёт запрос в одном из трёх описанных выше форматов. | Для каждого запроса find выведите в отдельной строке(файл txt) либо имя, либо «not found». |

# Схема алгоритма решения

1. Ввод данных.
2. Обработка команд.
3. При надобности - вывод информации.

# Полное описание реализованной функции

1. Устанавливаем лимит на количество контактов в телефонной книге.
2. Инициализируем пустой словарь book, который будет использоваться для хранения пар "номер телефона - имя".
3. Вводим n - количество команд.
4. Запускаем цикл, который будет выполняться n раз, чтобы обработать все запросы.
5. Считываем строку запроса и разбиваем ее на части, сохраняя в списке query.
6. Сохраняем первую часть запроса в переменной command, чтобы понять, какое действие выполнять (например, add, del, find).
7. Обработка команд:
   1. Команда add, сохраняем номер и имя из соответствующих частей списка query. Проверяем, не существует ли уже записи с данным номером и достигнут ли лимит по количеству контактов. Если условие истинно, выводим сообщение об ошибке и переходим к следующему запросу (пропускаем код дальше) - реализация усложнения на лимит контактов. Если лимит не достигнут или контакт уже существует, добавляем запись, устанавливая name для number в словаре book. Если введено существующее в словаре имя, выводится ошибка.
   2. Команда del, считываем номер. Если этот номер существует в book, удаляем соответствующую запись.
   3. Команда find, считываем номер. Если этот номер присутствует в book, выводим соответствующее имя. Если нет, выводим "not found".
   4. Отладочная команда - 0, выводим текущие записи в телефонной книге (весь словарь book).

# Листинг программы

def phone\_book(max\_contacts):

# Создаем словарь для хранения телефонной книги

book = {}

# Читаем количество запросов

n = int(input())

# Обрабатываем каждый запрос

for \_ in range(n):

query = input().split()

command = query[0]

if command == 'add':

number = query[1]

name = query[2]

# Проверяем, достигнут ли лимит по контактам

if number not in book and len(book) >= max\_contacts:

print("Error: contact limit reached")

continue

# Проверка на существование имени с другим номером

if name in book.values() and number not in book:

print("Error: name already used with a different number")

continue

else:

# Добавляем или обновляем запись

book[number] = name

elif command == 'del':

number = query[1]

# Удаляем запись, если она существует

if number in book:

del book[number]

elif command == 'find':

number = query[1]

# Ищем запись и выводим соответствующий результат

if number in book:

print(book[number])

else:

print("not found")

elif command == '0':

print(book)

# Устанавливаем лимит по количеству контактов

max\_contact\_limit = 3

# Запускаем телефонную книгу

phone\_book(max\_contact\_limit)

# 

# Результат выполнения программы.

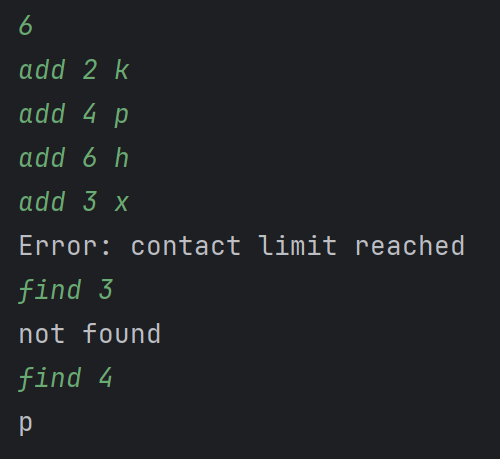


Рисунок 2.1 - 1-ый результат работы программы

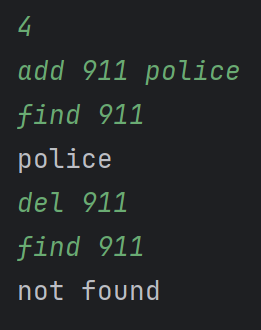


Рисунок 2.2. - 2-рой результат работы программы

# Выводы

В ходе выполнения лабораторной работы мной были освоены и изучены: понятие словарь; обработка последовательности запросов. Написанная программа была протестирована, полученный результат соответствует значению в примере.