МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ И ПРОГРАММНОЙ ИНЖЕНЕРИИ

КУРСОВАЯ РАБОТА (ПРОЕКТ)   
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| доцент |  |  |  | В.А. Матьяш |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ |
| **Разработка информационной системы «Обслуживание клиентов в бюро проката автомобилей»** |
| по дисциплине: СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ |
|  |
|  |

РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4134К |  |  |  | Н.С. Столяров |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2023

Содержание

[1. Задание на курсовой проект. 3](#_Toc135132166)

[2. Введение. 3](#_Toc135132167)

[3. Алгоритмы и структуры данных. 3](#_Toc135132168)

[4. Описание программы. 3](#_Toc135132169)

[5. Тестирование программы. 6](#_Toc135132170)

[6. Заключение. 9](#_Toc135132171)

[7. Список литературы. 9](#_Toc135132172)

[8. Приложение. 10](#_Toc135132173)

# 1. Задание на курсовой проект.

Задачей курсового проекта является разработка программы для заданной предметной области (“Регистрация постояльцев в гостинице ”)  
Расчет варианта: 2(Регистрация постояльцев в гостинице)  
Расчет метода хэширования: 2(Закрытое хэширование с квадратичным  
опробованием)  
Расчет метода сортировки: 5(Слиянием)  
Расчет вида списка: 1(Линейный двунаправленный)  
Расчет метода обхода дерева: 2(Прямой)  
Расчет алгоритма поиска слова в тексте: 0(Боуера и Мура)  
2. Введение.

Информационная система "Регистрация постояльцев в гостинице" предназначена для автоматизации процесса учета и управления данными о постояльцах, гостиничных номерах, а также информацией о вселении и выселении постояльцев. Она обеспечивает эффективное хранение, обработку и вывод данных, связанных с гостиничной деятельностью.

Цель данной системы заключается в облегчении работы персонала гостиницы, ускорении процесса регистрации постояльцев, предоставлении оперативной информации о занятости номеров, а также обеспечении возможности быстрого поиска нужных данных.

# 3. Алгоритмы и структуры данных.

1) Хеш-таблица:

Данные о постояльцах организованы в виде хеш-таблицы, где первичным ключом является номер паспорта. Для хеширования используется закрытое хеширование с квадратичным опробованием. Это обеспечивает быстрый доступ к данным по ключу.

2) АВЛ-дерево поиска:

Данные о гостиничных номерах организованы в виде АВЛ-дерева поиска, упорядоченного по номеру гостиничного номера. Эта структура данных обеспечивает эффективный поиск, вставку и удаление элементов.

3) Линейный двунаправленный список:

Данные о вселении или выселении постояльцев организованы в виде линейного двунаправленного списка, упорядоченного по номеру гостиничного номера. Этот список обеспечивает удобный доступ к данным о заселении и выселении, а также поддерживает операции добавления и удаления элементов.

4) Алгоритм поиска слова в тексте Бойера и Мура:

Для поиска гостиничных номеров по фрагментам "Оборудования" используется алгоритм Бойера и Мура. Он позволяет эффективно находить нужные записи, сравнивая фрагменты оборудования с полным перечнем оборудования каждого номера.

# 4. Описание программы.

**Руководство по использованию программы:**

Функция "Выход":

* + Описывает выход из программы.
  + При выборе этой опции программа завершается.

Функция "Регистрация постояльца в номер":

* + Позволяет зарегистрировать вселение постояльца в гостиничный номер.
  + Вводится информация о постояльце (например, ФИО, номер паспорта) и номере комнаты.
  + Проверяется доступность номера и сохраняются данные о вселении.

Функция "Выселение постояльца":

* + Позволяет зарегистрировать выселение постояльца из гостиничного номера.
  + Вводится информация о постояльце или номере комнаты для выселения.
  + Проверяется наличие данных о вселении и производится удаление этих данных.

Функция "Список заселенных постояльцев":

* + Выводит на экран список всех заселенных постояльцев с информацией о них и номерах комнат, в которых они проживают.

Функция "Регистрация постояльца":

* + Позволяет зарегистрировать нового постояльца в базе данных.
  + Вводится информация о постояльце, такая как ФИО, номер паспорта и другие данные.
  + Добавленные данные сохраняются в базе данных.

Функция "Удаление данных постояльца":

* + Позволяет удалить данные о постояльце из базы данных.
  + Вводится информация о постояльце, например, номер паспорта, для идентификации удаляемой записи.
  + Производится удаление данных о постояльце и связанных с ним операциях вселения и выселения.

Функция "Список постояльцев":

* + Выводит на экран список всех зарегистрированных постояльцев с их данными.

Функция "Очистить список постояльцев":

* + Удаляет все данные о зарегистрированных постояльцах из базы данных.
  + После выполнения этой операции список постояльцев становится пустым.

Функция "Поиск постояльца по номеру паспорта":

* + Позволяет найти информацию о постояльце по его номеру паспорта.
  + Вводится номер паспорта, и если такой постоялец найден, выводится его информация.

Функция "Поиск постояльца по ФИО":

* + Позволяет найти информацию о постояльце по его ФИО (имени, фамилии).
  + Вводится ФИО постояльца или его часть, и если такой постоялец найден, выводится его информация.

Функция "Добавить гостиничный номер":

* + Позволяет добавить новый гостиничный номер в базу данных.
  + Вводится информация о номере, например, его номер и оборудование.
  + Добавленный номер сохраняется в базе данных.

Функция "Удалить гостиничный номер":

* + Позволяет удалить гостиничный номер из базы данных.
  + Вводится номер комнаты для удаления.
  + Производится удаление данных о номере и связанных с ним операциях вселения и выселения.

Функция "Вывести список гостиничных номеров":

* + Выводит на экран список всех гостиничных номеров с информацией о них.

Функция "Очистить все данные о гостиничных номерах":

* + Удаляет все данные о гостиничных номерах из базы данных.
  + После выполнения этой операции список гостиничных номеров становится пустым.

Функция "Поиск гостиничного номера по номеру":

* + Позволяет найти информацию о гостиничном номере по его номеру.
  + Вводится номер комнаты, и если такой номер найден, выводится его информация.

Функция "Поиск гостиничного номера по оборудованию":

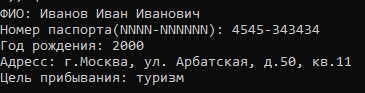
* + Позволяет найти информацию о гостиничных номерах по фрагменту оборудования.
  + Вводится фрагмент оборудования, и если такие номера найдены, выводится их информация.

# 5. Тестирование программы.

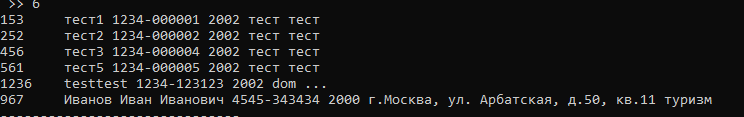
Исходные данные для тестовых прогонов программы:

|  |
| --- |
| {  "roomers": {  "153": {  "name": "\u0442\u0435\u0441\u04421",  "pasport": "1234-000001",  "birth": 2002,  "adress": "\u0442\u0435\u0441\u0442",  "purpose": "\u0442\u0435\u0441\u0442"  },  "252": {  "name": "\u0442\u0435\u0441\u04422",  "pasport": "1234-000002",  "birth": 2002,  "adress": "\u0442\u0435\u0441\u0442",  "purpose": "\u0442\u0435\u0441\u0442"  },  "456": {  "name": "\u0442\u0435\u0441\u04423",  "pasport": "1234-000004",  "birth": 2002,  "adress": "\u0442\u0435\u0441\u0442",  "purpose": "\u0442\u0435\u0441\u0442"  },  "561": {  "name": "\u0442\u0435\u0441\u04425",  "pasport": "1234-000005",  "birth": 2002,  "adress": "\u0442\u0435\u0441\u0442",  "purpose": "\u0442\u0435\u0441\u0442"  },  "1236": {  "name": "testtest",  "pasport": "1234-123123",  "birth": 2002,  "adress": "dom",  "purpose": "..."  }  },  "hotel\_rooms": [  {  "number": "\u041f001",  "count\_seats": 4,  "count\_rooms": 2,  "bathroom": **true**,  "equipment": "\u043e\u043a\u043d\u043e"  },  {  "number": "\u041f003",  "count\_seats": 3,  "count\_rooms": 4,  "bathroom": **false**,  "equipment": "\u043a\u0440\u043e\u0432\u0430\u0442\u044c \u0445\u043e\u043b\u043e\u0434\u0438\u043b\u044c\u043d\u0438\u043a \u0432\u0435\u0434\u0440\u043e"  }  ],  "move\_roomer": [  {  "pasport": "1234-000001",  "number": "\u041f001",  "date\_in": "2016",  "date\_out": "2025"  },  {  "pasport": "1234-000002",  "number": "\u041f001",  "date\_in": "2016",  "date\_out": "2025"  },  {  "pasport": "1234-000003",  "number": "\u041f001",  "date\_in": "2016",  "date\_out": "2025"  },  {  "pasport": "1234-000004",  "number": "\u041f001",  "date\_in": "2016",  "date\_out": "2025"  }  ]  } |

Действия, которые проводились с базой:  
1) Зарегистрировали нового постояльца.



2) Вывели список постояльцев.

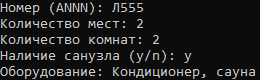


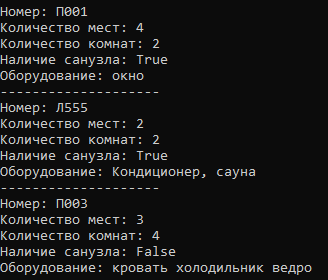
3) Удалили данные постояльца.  


4) Нашли постояльца по номеру паспорта.

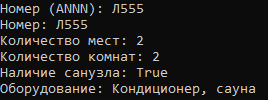


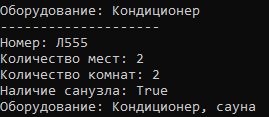
5) Нашли постояльца по ФИО.

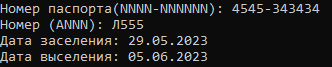
6) Добавили новый номер.  


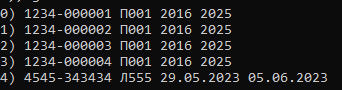
7) Вывели список всех номеров.  


8) Удалили номер.  


9) Нашли гостиничный номер по номеру.  


10) Нашли гостиничный номер по оборудованию.  


11) Заселили клиента в номер.  


12) Вывели список заселенных номеров.  


13) Выселили постояльца.



# 6. Заключение.

Реализация программы была выполнена с использованием вычислительных машин, которые обеспечивают хранение данных и выполнение операций. Корректное функционирование программы зависит от правильного ввода данных и соблюдения инструкций программы.

В результате разработки данной информационной системы, достигнута цель создания эффективного и удобного инструмента для регистрации и учета постояльцев в гостинице, что способствует более эффективному управлению гостиничным бизнесом и улучшению обслуживания гостей.

В результате выполнения данного курсового проекта были получены навыки разработки программного обеспечения, применения алгоритмов и структур данных, а также улучшены навыки в программировании и организации проекта.

# 7. Список литературы.

1)Алгоритмы и структуры данных: учеб. пособие / В. А. Матьяш, С. А. Рогачев. – СПб.: ГУАП, 2021  
2) Структуры и алгоритмы обработки данных / Павлов, Первова:, Лань,2021  
3) Алгоритмы/ Вазирани Умеш, Дасгупта Санджой

# 8. Приложение.

**8.1** **Листинг программы:**

menu.py

|  |
| --- |
| **import** sys  **from** sys **import** platform  **import** os  **def** clear\_screen**():**  **if** platform **==** "linux" **or** platform **==** "linux2"**:**  os**.**system**(**"clear"**)**  **elif** platform **==** "darwin"**:**  os**.**system**(**"clear"**)**  **elif** platform **==** "win32"**:**  os**.**system**(**"cls"**)**  **def** check\_format**(str,** template**):**  **if** **len(str)** **!=** **len(**template**):**  **return** **False**  **for** i **in** **range(len(str)):**  **if** template**[**i**]** **==** "N" **and** **not** **str[**i**].**isdigit**():**  **return** **False**  **if** template**[**i**]** **==** "A" **and** **not** **(str[**i**]** **in** **[**"Л"**,** "П"**,** "О"**,** "М"**]):**  **return** **False**  **if** template**[**i**]** **==** "-" **and** **not** **(str[**i**]** **==** "-"**):**  **return** **False**  **return** **True**  **def** read\_bool**(**promt**=**""**):**  **while** **True:**  read **=** **input(**promt **+** " (y/n): "**)**  **if** read **in** **[**"y"**,** "Y"**,** "n"**,** "N"**,** "д"**,** "Д"**,** "н"**,** "Н"**]:**  **return** **(**read **in** **[**"y"**,** "Y"**,** "д"**,** "Д"**])**  **class** **Menu:**  **def** \_\_init\_\_**(**self**,** elements**,** function**=None):**  self**.**elements **=** elements  self**.**function **=** function  **def** run**(**self**):**  #clear\_screen()  **while** **True:**  **print(**"-" **\*** 30**)**  **for** i **in** **range(len(**self**.**elements**)):**  **print(**"%d) %s" **%** **(**i**,** self**.**elements**[**i**][**0**]))**  **try:**  menu\_input **=** **int(input(**"\n >> "**))**  #clear\_screen()  self**.**elements**[**menu\_input**][**1**]()**  **if** self**.**function **!=** **None:**  self**.**function**()**  **except** **Exception** **as** e**:**  **print(**"#" **\*** 30**)**  **print(**"WARNING"**)**  exc\_type**,** exc\_obj**,** exc\_tb **=** sys**.**exc\_info**()**  fname **=** os**.**path**.**split**(**exc\_tb**.**tb\_frame**.**f\_code**.**co\_filename**)[**1**]**  **print(**exc\_type**,** fname**,** exc\_tb**.**tb\_lineno**)**  **print(**e**)**  **print(**"#" **\*** 30**)** |

avltree.py

|  |
| --- |
| **from** search **import** **\***  **class** **AVLTree:**  **def** \_\_init\_\_**(**self**):**  self**.**root **=** **None**  self**.**client **=** **None**  self**.**left **=** **None**  self**.**right **=** **None**  self**.**height **=** 1  **def** clear**(**self**):**  self**.**root **=** **None**  self**.**client **=** **None**  self**.**left **=** **None**  self**.**right **=** **None**  **def** insert**(**self**,** node**,** client**):**  **if** node **==** **None:**  node **=** AVLTree**()**  node**.**client **=** **None**  **if** node**.**client **==** **None:**  node**.**client **=** client  **return** node  **elif** client**.**number **<** node**.**client**.**number**:**  node**.**left **=** self**.**insert**(**node**.**left**,** client**)**  **else:**  node**.**right **=** self**.**insert**(**node**.**right**,** client**)**  node**.**height **=** 1 **+** **max(**self**.**get\_height**(**node**.**left**),** self**.**get\_height**(**node**.**right**))**  **return** node  **def** left\_rotate**(**self**,** node**):**  right **=** node**.**right  left **=** right**.**left  right**.**left **=** node  node**.**right **=** left  node**.**height **=** 1 **+** **max(**self**.**get\_height**(**node**.**left**),** self**.**get\_height**(**node**.**right**))**  right**.**height **=** 1 **+** **max(**self**.**get\_height**(**right**.**left**),** self**.**get\_height**(**right**.**right**))**  **return** right  **def** right\_rotate**(**self**,** node**):**  left **=** node**.**left  right **=** left**.**right  left**.**right **=** node  node**.**left **=** right  node**.**height **=** 1 **+** **max(**self**.**get\_height**(**node**.**left**),** self**.**get\_height**(**node**.**right**))**  left**.**height **=** 1 **+** **max(**self**.**get\_height**(**left**.**left**),** self**.**get\_height**(**left**.**right**))**  **return** left  **def** get\_height**(**self**,** node**):**  **if** node **is** **None:**  **return** 0  **return** node**.**height  **def** get\_balance**(**self**,** node**):**  **if** node **is** **None:**  **return** 0  **return** self**.**get\_height**(**node**.**left**)** **-** self**.**get\_height**(**node**.**right**)**  **def** search**(**self**,** node**,** number**):**  **if** node **is** **None:**  **return** **None**  **if** number **==** node**.**client**.**number**:**  **return** node**.**client  **if** number **<** node**.**client**.**number**:**  **return** self**.**search**(**node**.**left**,** number**)**  **return** self**.**search**(**node**.**right**,** number**)**  **def** pre\_order**(**self**,** node**):**  **if** **(**node**):**  **print(**"-" **\*** 20**)**  **if** node**.**client **!=** **None:**  node**.**client**.print()**  self**.**pre\_order**(**node**.**left**)**  self**.**pre\_order**(**node**.**right**)**  **def** pre\_order\_return**(**self**,** node**):**  out **=** **[]**  **if** **(**node**):**  **if** node**.**client **!=** **None:**  out**.**append**(**node**.**client**)**  out **+=** self**.**pre\_order\_return**(**node**.**left**)**  out **+=** self**.**pre\_order\_return**(**node**.**right**)**  **return** out  ################################################################################  # поиск по заданию  **def** search\_by\_number**(**self**,** number**):**  current **=** self  **while** **True:**  **if** current **==** **None:**  **return** **None**  **if** current**.**client**.**number **==** number**:**  **return** current**.**client  **elif** number **<** current**.**client**.**number**:**  current **=** current**.**left  **else:**  current **=** current**.**right  **def** search\_by\_equipment**(**self**,** equipment**,** current**=None):**  **if** current **==** **None:**  current **=** self  result **=** **[]**  **if** boyerMurSearch**(**equipment**,** current**.**client**.**equipment**):**  result**.**append**(**current**.**client**)**  **if** current**.**left **!=** **None:**  result **+=** self**.**search\_by\_equipment**(**equipment**,** current**.**left**)**  **if** current**.**right **!=** **None:**  result **+=** self**.**search\_by\_equipment**(**equipment**,** current**.**right**)**  **return** result  **def** delete\_by\_number**(**self**,** number**):**  current **=** self  go\_left **=** **False**  prev **=** **None**  **while** **True:**  **if** current **==** **None:**  **return** **False**  **print(**current**.**client**.**number**)**  **if** current**.**client**.**number **==** number**:**  **if** prev **!=** **None:**  **if** go\_left**:**  prev**.**left **=** **None**  **else:**  prev**.**right **=** **None**  **else:**  prev **=** self  left **=** current**.**left  right **=** current**.**right  **if** left **!=** **None:**  self**.**insert**(**prev**,** left**.**client**)**  **if** right **!=** **None:**  self**.**insert**(**prev**,** right**.**client**)**  **return** **True**  **else:**  prev **=** current  **if** number **<** current**.**client**.**number**:**  current **=** current**.**left  go\_left **=** **True**  **else:**  current **=** current**.**right  go\_left **=** **False** |

file.py

|  |
| --- |
| **import** os  **import** json  **def** save\_dict**(dict,** name**):**  json**.**dump**(dict,** **open(str(**name**)** **+** '.json'**,**'w'**),** indent**=**2**)**  **def** read\_dict**(**name**):**  **with** **open(str(**name**)** **+** '.json'**,** encoding**=**'utf-8'**)** **as** fh**:**  data **=** json**.**load**(**fh**)**  **return** data  **class** **Data\_file:**  **def** \_\_init\_\_**(**self**,** file\_name**=**"data"**):**  self**.**file\_name **=** file\_name  self**.**roomers **=** **{}**  self**.**hotel\_rooms **=** **[]**  self**.**move\_roomer **=** **[]**  self**.**read**()**  **def** clear**(**self**):**  self**.**roomers **=** **{}**  self**.**hotel\_rooms **=** **[]**  self**.**move\_roomer **=** **[]**  **def** save**(**self**):**  data **=** **{**  "roomers"**:** self**.**roomers**,**  "hotel\_rooms"**:** self**.**hotel\_rooms**,**  "move\_roomer"**:** self**.**move\_roomer  **}**  save\_dict**(**data**,** self**.**file\_name**)**  **def** read**(**self**):**  **if** **not** os**.**path**.**exists**(**self**.**file\_name **+** '.json'**):**  self**.**save**()**  self**.**read**()**  **return**  data **=** read\_dict**(**self**.**file\_name**)**  self**.**roomers **=** data**[**"roomers"**]**  self**.**hotel\_rooms **=** data**[**"hotel\_rooms"**]**  self**.**move\_roomer **=** data**[**"move\_roomer"**]**  **def** get\_all**(**self**):**  **return** self**.**roomers**,** self**.**hotel\_rooms**,** self**.**move\_roomer |

hotel\_rooms.py

|  |
| --- |
| # информация о гостиночных номерах  **class** **Room:**  **def** \_\_init\_\_**(**self**,** number**,** count\_seats**,** count\_rooms**,** bathroom**,** equipment**):**  self**.**number **=** number # строка формата ANNN, где A – буква,  # обозначающая тип номера (Л – люкс, П – полулюкс,  # О – одноместный, М – многоместный); NNN – порядковый номер (цифры)  self**.**count\_seats **=** count\_seats # Количество мест  self**.**count\_rooms **=** count\_rooms # Количество комнат  self**.**bathroom **=** bathroom # Наличие санузла (bool)  self**.**equipment **=** equipment # Оборудование  **def** print**(**self**):**  **print(**"Номер: %s\nКоличество мест: %d\nКоличество комнат: %d\nНаличие санузла: %s\nОборудование: %s" **%** **(**  self**.**number**,**  self**.**count\_seats**,**  self**.**count\_rooms**,**  **(**"True" **if** self**.**bathroom **else** "False"**),**  self**.**equipment  **))** |

main.py

|  |
| --- |
| **import** datetime  **from** avltree **import** **\***  **from** roomers **import** **\***  **from** hotel\_rooms **import** **\***  **from** move\_roomers **import** **\***  **from** file **import** **\***  **from** sort **import** **\***  **from** menu **import** **\***  ################################################################################  **def** get\_current\_year**():**  **return** datetime**.**datetime**.**now**().**year  ################################################################################  data\_file **=** Data\_file**(**"data"**)**  move\_list **=** **None**  tree **=** AVLTree**()**  hash\_table **=** HashTable**()**  ################################################################################  **def** sort\_move\_list**():**  **global** move\_list  arr **=** move\_list**.**get\_all**()**  move\_list **=** **None**  merge\_sort**(**arr**,** 0**,** **len(**arr**))**  **for** move\_roomer **in** arr**:**  **if** move\_list **==** **None:**  move\_list **=** Element\_list**(**move\_roomer**)**  **else:**  move\_list**.**add**(**move\_roomer**)**  ################################################################################  **def** read\_file**():**  **global** move\_list  **global** tree  **for** key **in** data\_file**.**roomers**:**  hash\_table**.**add**(**Roomer**(\*\***data\_file**.**roomers**[**key**]))**  **for** el **in** data\_file**.**hotel\_rooms**:**  tree**.**insert**(**tree**,** Room**(\*\***el**))**  **for** el **in** data\_file**.**move\_roomer**:**  move\_roomer **=** Move\_roomer**(\*\***el**)**  **if** move\_list **==** **None:**  move\_list **=** Element\_list**(**move\_roomer**)**  **else:**  move\_list**.**add**(**move\_roomer**)**  **def** save\_to\_file**():**  data\_file**.**clear**()**  **for** key **in** hash\_table**.**hash\_dict**:**  data **=** **{}**  data**[**"name"**]** **=** hash\_table**.**hash\_dict**[**key**].**name  data**[**"pasport"**]** **=** hash\_table**.**hash\_dict**[**key**].**pasport  data**[**"birth"**]** **=** hash\_table**.**hash\_dict**[**key**].**birth  data**[**"adress"**]** **=** hash\_table**.**hash\_dict**[**key**].**adress  data**[**"purpose"**]** **=** hash\_table**.**hash\_dict**[**key**].**purpose  data\_file**.**roomers**[**key**]** **=** data  tree\_return **=** tree**.**pre\_order\_return**(**tree**)**  **for** el **in** tree\_return**:**  data **=** **{}**  data**[**"number"**]** **=** el**.**number  data**[**"count\_seats"**]** **=** el**.**count\_seats  data**[**"count\_rooms"**]** **=** el**.**count\_rooms  data**[**"bathroom"**]** **=** el**.**bathroom  data**[**"equipment"**]** **=** el**.**equipment  data\_file**.**hotel\_rooms**.**append**(**data**)**  **if** move\_list **!=** **None:**  move\_list\_out **=** move\_list**.**get\_all**()**  **for** el **in** move\_list\_out**:**  data **=** **{}**  data**[**"pasport"**]** **=** el**.**pasport  data**[**"number"**]** **=** el**.**number  data**[**"date\_in"**]** **=** el**.**date\_in  data**[**"date\_out"**]** **=** el**.**date\_out  data\_file**.**move\_roomer**.**append**(**data**)**  data\_file**.**save**()**  read\_file**()**  ################################################################################  **def** add\_roomer**():**  **global** hash\_table  data **=** **{}**  data**[**"name"**]** **=** **input(**"ФИО: "**)**  data**[**"pasport"**]** **=** **input(**"Номер паспорта(NNNN-NNNNNN): "**)**  **if** **not** check\_format**(**data**[**"pasport"**],** "NNNN-NNNNNN"**):**  **print(**"Строка не соответствует формату"**)**  **return**  data**[**"birth"**]** **=** **int(input(**"Год рождения: "**))**  **if** **(**data**[**"birth"**]** **<** 1900 **and** data**[**"birth"**]** **<** get\_current\_year**()):**  **print(**"Дата не корректна"**)**  **return**  data**[**"adress"**]** **=** **input(**"Адресс: "**)**  data**[**"purpose"**]** **=** **input(**"Цель прибывания: "**)**  hash\_table**.**add**(**Roomer**(\*\***data**))**  **def** delete\_roomer**():**  pasport **=** **input(**"Номер паспорта постаяльца(NNNN-NNNNNN): "**)**  **if** **not** check\_format**(**pasport**,** "NNNN-NNNNNN"**):**  **print(**"Строка не соответствует формату"**)**  **return**  **hash,** roomer **=** hash\_table**.**find\_by\_pasport**(**pasport**)**  **if** **hash** **==** **None:**  **print(**"Такого постаяльца нет"**)**  **return**  **id** **=** move\_list**.**search\_by\_pasport**(**pasport**)**  **if** **id** **!=** **None:**  move\_list**.**delete\_element**(id)**  hash\_table**.**remove**(hash)**  **def** show\_roomers**():**  hash\_table**.**print\_table**()**  **def** clear\_roomers**():**  hash\_table**.**clear\_table**()**  **def** find\_roomer\_by\_passport**():**  pasport **=** **input(**"Номер паспорта постаяльца(NNNN-NNNNNN): "**)**  **if** **not** check\_format**(**pasport**,** "NNNN-NNNNNN"**):**  **print(**"Строка не соответствует формату"**)**  **return**  **hash,** roomer **=** hash\_table**.**find\_by\_pasport**(**pasport**)**  **if** **hash** **==** **None:**  **print(**"Такого постаяльца нет"**)**  **return**  roomer**.print()**  **def** find\_roomer\_by\_name**():**  name **=** **input(**"ФИО постаяльца: "**)**  **hash,** roomer **=** hash\_table**.**find\_by\_name**(**name**)**  **if** **hash** **==** **None:**  **print(**"Такого постаяльца нет"**)**  **return**  roomer**.print()**  ################################################################################  **def** add\_hotel\_room**():**  **global** tree  data **=** **{}**  data**[**"number"**]** **=** **input(**"Номер (ANNN): "**)**  **if** **not** check\_format**(**data**[**"number"**],** "ANNN"**):**  **print(**"Строка не соответствует формату"**)**  **return**  data**[**"count\_seats"**]** **=** **abs(int(input(**"Количество мест: "**)))**  data**[**"count\_rooms"**]** **=** **abs(int(input(**"Количество комнат: "**)))**  data**[**"bathroom"**]** **=** read\_bool**(**"Наличие санузла"**)**  data**[**"equipment"**]** **=** **input(**"Оборудование: "**)**  room **=** Room**(\*\***data**)**  tree**.**insert**(**tree**,** room**)**  **def** print\_hotel\_room**():**  tree**.**pre\_order**(**tree**)**  **def** delte\_hotel\_room**():**  number **=** **input(**"Номер (ANNN): "**)**  **if** **not** check\_format**(**number**,** "ANNN"**):**  **print(**"Строка не соответствует формату"**)**  **return**  **if** **not** tree**.**delete\_by\_number**(**number**):**  **print(**"Такого номера не существует"**)**  roomers **=** move\_list**.**search\_by\_number**(**number**)**  **for** i **in** **range(len(**roomers**)-**1**,** **-**1**,** **-**1**):**  **id** **=** move\_list**.**search\_by\_pasport**(**roomers**[**i**].**pasport**)**  move\_list**.**delete\_element**(id)**  **def** clear\_hotel\_rooms**():**  tree**.**clear**()**  **def** search\_hotel\_room\_by\_number**():**  number **=** **input(**"Номер (ANNN): "**)**  **if** **not** check\_format**(**number**,** "ANNN"**):**  **print(**"Строка не соответствует формату"**)**  **return**  room **=** tree**.**search\_by\_number**(**number**)**  **if** room **==** **None:**  **print(**"Такого номера нет"**)**  **return**  room**.print()**  roomers **=** move\_list**.**search\_by\_number**(**number**)**  **for** roomer **in** roomers**:**  **hash,** roomer **=** hash\_table**.**find\_by\_pasport**(**roomer**.**pasport**)**  **if** **hash** **!=** **None:**  **print()**  roomer**.print()**  **def** search\_hotel\_room\_by\_equipment**():**  equipment **=** **input(**"Оборудование: "**)**  arr **=** tree**.**search\_by\_equipment**(**equipment**)**  **for** el **in** arr**:**  **print(**"-" **\*** 20**)**  el**.print()**  roomers **=** move\_list**.**search\_by\_number**(**el**.**number**)**  **for** roomer **in** roomers**:**  **hash,** roomer **=** hash\_table**.**find\_by\_pasport**(**roomer**.**pasport**)**  **if** **hash** **!=** **None:**  **print()**  roomer**.print()**  ################################################################################  **def** add\_move**():**  **global** move\_list  data **=** **{}**  data**[**"pasport"**]** **=** **input(**"Номер паспорта(NNNN-NNNNNN): "**)**  **if** **not** check\_format**(**data**[**"pasport"**],** "NNNN-NNNNNN"**):**  **print(**"Строка не соответствует формату"**)**  **return**  **hash,** roomer **=** hash\_table**.**find\_by\_pasport**(**data**[**"pasport"**])**  **if** **hash** **==** **None:**  **print(**"Такого постаяльца нет"**)**  **return**  data**[**"number"**]** **=** **input(**"Номер (ANNN): "**)**  **if** **not** check\_format**(**data**[**"number"**],** "ANNN"**):**  **print(**"Строка не соответствует формату"**)**  **return**  room **=** tree**.**search\_by\_number**(**data**[**"number"**])**  **if** room **==** **None:**  **print(**"Такого номера нет"**)**  **return**  **if** move\_list **!=** **None:**  roomers **=** move\_list**.**search\_by\_number**(**data**[**"number"**])**  **if** **len(**roomers**)** **>=** room**.**count\_seats**:**  **print(**"В номере нет мест"**)**  **return**  data**[**"date\_in"**]** **=** **input(**"Дата заселения: "**)**  data**[**"date\_out"**]** **=** **input(**"Дата выселения: "**)**  move\_roomer **=** Move\_roomer**(\*\***data**)**  **if** move\_list **==** **None:**  move\_list **=** Element\_list**(**move\_roomer**)**  **else:**  move\_list**.**add**(**move\_roomer**)**  # sort\_move\_list()  **def** delete\_move**():**  passport **=** **input(**"Номер паспорта(NNNN-NNNNNN): "**)**  **if** **not** check\_format**(**passport**,** "NNNN-NNNNNN"**):**  **print(**"Строка не соответствует формату"**)**  **return**  **id** **=** move\_list**.**search\_by\_pasport**(**passport**)**  **if** **id** **==** **None:**  **print(**"Такого постаяльца нет"**)**  **return**  move\_list**.**delete\_element**(id)**  sort\_move\_list**()**  **def** print\_move**():**  **global** move\_list  **for** i **in** **range(**move\_list**.**get\_length**()):**  **print(**"%d) %s %s %s %s" **%** **(**  i**,**  move\_list**.**get\_current\_item**(**i**).**item**.**pasport**,**  move\_list**.**get\_current\_item**(**i**).**item**.**number**,**  move\_list**.**get\_current\_item**(**i**).**item**.**date\_in**,**  move\_list**.**get\_current\_item**(**i**).**item**.**date\_out  **))**  ################################################################################  menu **=** Menu**([**  **[**"Выход\n"**,** **exit],**  **[**"Регистрация постаяльца в номер"**,** add\_move**],**  **[**"Выселение постаяльца"**,** delete\_move**],**  **[**"Список заселённых постаяльцев\n"**,** print\_move**],**  **[**"Регистрация постаяльца"**,** add\_roomer**],**  **[**"Удаление данных постаяльца"**,** delete\_roomer**],**  **[**"Список постаяльцев"**,** show\_roomers**],**  **[**"Очистить список постаяльцев"**,** clear\_roomers**],**  **[**"Поиск постаяльца по номеру паспорта"**,** find\_roomer\_by\_passport**],**  **[**"Поиск постаяльца по ФИО\n"**,** find\_roomer\_by\_name**],**  **[**"Добавить гостиночный номер"**,** add\_hotel\_room**],**  **[**"Удалить гостиночный номер"**,** delte\_hotel\_room**],**  **[**"Вывести список гостиночных номеров"**,** print\_hotel\_room**],**  **[**"Очистить все данные о гостиночных номерах"**,** clear\_hotel\_rooms**],**  **[**"Поиск гостиночного номера по номеру"**,** search\_hotel\_room\_by\_number**],**  **[**"Поиск гостиночного номера по оборудованию"**,** search\_hotel\_room\_by\_equipment**]**  **],** save\_to\_file**)**  menu**.**run**()** |

move\_roomers.py

|  |
| --- |
| # вселение и выселение постояльцев  **class** **Move\_roomer():**  **def** \_\_init\_\_**(**self**,** pasport**,** number**,** date\_in**,** date\_out**):**  self**.**pasport **=** pasport  self**.**number **=** number  self**.**date\_in **=** date\_in  self**.**date\_out **=** date\_out  **class** **Element\_list:**  **def** \_\_init\_\_**(**self**,** item**,** head**=None):**  self**.**item **=** item  self**.next** **=** **None**  self**.**previous **=** **None**  self**.**head **=** **(**self **if** **(**head **==** **None)** **else** head**)**  **def** add**(**self**,** item**):**  **next** **=** self**.**head  **while** **(next.next** **!=** **None):**  **next** **=** **next.next**  **next.next** **=** Element\_list**(**item**,** self**.**head**)**  **next.next.**previous **=** **next.next**  **return** **next.next**  **def** get\_item**(**self**):**  **return** self**.**item  **def** get\_head**(**self**):**  **return** self**.**head  **def** get\_all**(**self**):**  result **=** **[]**  head **=** self**.**head  **while** **(**head **!=** **None):**  result**.**append**(**head**.**item**)**  head **=** head**.next**  **return** result  **def** get\_current\_item**(**self**,** **id):**  head **=** self**.**head  **for** \_ **in** **range(id):**  head **=** head**.next**  **return** head  **def** delete\_element**(**self**,** **id):**  **if** **(id** **<** 0 **or** **id** **>** **(**self**.**get\_length**()** **-** 1**)):**  **return** **False**  **if** **id** **==** 0**:**  self**.**head **=** self**.**head**.next**  **return** **True**  elem\_curr **=** self**.**head  elem\_prev **=** **None**  elem\_next **=** **None**  **for** i **in** **range(id):**  elem\_prev **=** elem\_curr  elem\_curr **=** elem\_curr**.next**  elem\_next **=** elem\_curr**.next**  elem\_prev**.next** **=** elem\_next  elem\_next**.**previous **=** elem\_prev  **return** **True**  **def** get\_length**(**self**):**  head **=** self**.**head  count **=** 0  **while** **(**head **!=** **None):**  head **=** head**.next**  count **+=** 1  **return** count  ################################################################################  **def** search\_by\_pasport**(**self**,** pasport**):**  head **=** self**.**head  count **=** 0  **while** **(**head **!=** **None):**  **if** self**.**item**.**pasport **==** pasport**:**  **return** count  head **=** head**.next**  count **+=** 1  **return** **None**  **def** search\_by\_number**(**self**,** number**):**  head **=** self**.**head  result **=** **[]**  **while** **(**head **!=** **None):**  **if** head**.**item**.**number **==** number**:**  result**.**append**(**head**.**item**)**  head **=** head**.next**  **return** result |

roomers.py

|  |
| --- |
| # информация о постояльцах  **import** random  **class** **Roomer:**  **def** \_\_init\_\_**(**self**,** pasport**,** birth**,** adress**,** purpose**,** name**):**  self**.**pasport **=** pasport # (строка) номер паспорта формата NNNN-NNNNNN (N - цифра)  self**.**birth **=** birth # (число) год рождения  self**.**adress **=** adress # (строка) адресс  self**.**purpose **=** purpose # (строка) цель прибывания  self**.**name **=** name  **def** print**(**self**):**  **print(**"%s %s %s %s %s" **%** **(**  self**.**name**,**  self**.**pasport**,**  self**.**birth**,**  self**.**adress**,**  self**.**purpose  **))**  **class** **HashTable:**  **def** \_\_init\_\_**(**self**,** template\_key**=**"NNNN-NNNNNN"**,** count\_sigments**=**2000**):**  self**.**template\_key **=** template\_key # формат ключа (N - число, C - символ)  self**.**count\_sigments **=** count\_sigments  self**.**hash\_dict **=** **{}**  **def** generate\_hash**(**self**,** key**):**  out **=** 1  **for** el **in** key**:**  out **+=** **ord(**el**)** **\*\*** 2  **return** **int(**out **%** self**.**count\_sigments**)**  **def** add**(**self**,** roomer**):**  **hash** **=** self**.**generate\_hash**(**roomer**.**pasport**)**  count\_collision **=** 1  **while** **True:**  **if** **hash** **in** self**.**hash\_dict**:**  **hash** **+=** count\_collision  count\_collision **\*=** 2  **else:**  self**.**hash\_dict**[hash]** **=** roomer  **break**  **def** get\_by\_hash**(**self**,** **hash):**  **return** self**.**hash\_dict**[hash]**  **def** remove**(**self**,** **hash):**  **return** self**.**hash\_dict**.**pop**(hash)**  **def** get\_by\_hash**(**self**,** **hash):**  **return** self**.**hash\_dict**[hash]**  **def** clear\_table**(**self**):**  self**.**hash\_dict **=** **{}**  **def** print\_table**(**self**):**  **for** el **in** self**.**hash\_dict**:**  **print(**"%d\t" **%** el**,** end**=**""**)**  self**.**hash\_dict**[**el**].print()**  ################################################################################  # поиск по заданию  **def** find\_by\_pasport**(**self**,** pasport**):**  **for** **hash** **in** self**.**hash\_dict**:**  **if** pasport **==** self**.**hash\_dict**[hash].**pasport**:**  **return** **hash,** self**.**hash\_dict**[hash]**  **return** **None,** **None**  **def** find\_by\_name**(**self**,** name**):**  **for** **hash** **in** self**.**hash\_dict**:**  **if** name **==** self**.**hash\_dict**[hash].**name**:**  **return** **hash,** self**.**hash\_dict**[hash]**  **return** **None,** **None** |

search.py

|  |
| --- |
| **def** boyerMurSearch**(**str1**,** str2**):**  t **=** str1  # Этап 1: формирование таблицы смещений  S **=** **set()** # уникальные символы в образе  M **=** **len(**t**)** # число символов в образе  d **=** **{}** # словарь смещений  **for** i **in** **range(**M**-**2**,** **-**1**,** **-**1**):** # итерации с предпоследнего символа  **if** t**[**i**]** **not** **in** S**:** # если символ еще не добавлен в таблицу  d**[**t**[**i**]]** **=** M**-**i**-**1  S**.**add**(**t**[**i**])**  **if** t**[**M**-**1**]** **not** **in** S**:** # отдельно формируем последний символ  d**[**t**[**M**-**1**]]** **=** M  d**[**'\*'**]** **=** M # смещения для прочих символов  # print(d)  # Этап 2: поиск образа в строке  a **=** str2  N **=** **len(**a**)**  **if** N **>=** M**:**  i **=** M**-**1 # счетчик проверяемого символа в строке  **while(**i **<** N**):**  k **=** 0  j **=** 0  flBreak **=** **False**  **for** j **in** **range(**M**-**1**,** **-**1**,** **-**1**):**  **if** a**[**i**-**k**]** **!=** t**[**j**]:**  **if** j **==** M**-**1**:**  off **=** d**[**a**[**i**]]** **if** d**.**get**(**a**[**i**],** **False)** **else** d**[**'\*'**]** # смещение, если не равен последний символ образа  **else:**  off **=** d**[**t**[**j**]]** # смещение, если не равен не последний символ образа  i **+=** off # смещение счетчика строки  flBreak **=** **True** # если несовпадение символа, то flBreak = True  **break**  k **+=** 1 # смещение для сравниваемого символа в строке  **if** **not** flBreak**:** # если дошли до начала образа, значит, все его символы совпали  # print(f"образ найден по индексу {i-k+1}")  **return** **True**  **break**  **else:**  # print("образ не найден")  **return** **False**  **else:**  # print("образ не найден")  **return** **False** |

sort.py

|  |
| --- |
| **def** merge\_sort**(**alist**,** start**,** end**):**  **if** end **-** start **>** 1**:**  mid **=** **(**start **+** end**)//**2  merge\_sort**(**alist**,** start**,** mid**)**  merge\_sort**(**alist**,** mid**,** end**)**  merge\_list**(**alist**,** start**,** mid**,** end**)**  **def** merge\_list**(**alist**,** start**,** mid**,** end**):**  left **=** alist**[**start**:**mid**]**  right **=** alist**[**mid**:**end**]**  k **=** start  i **=** 0  j **=** 0  **while** **(**start **+** i **<** mid **and** mid **+** j **<** end**):**  **if** **(**left**[**i**].**pasport **<=** right**[**j**].**pasport**):**  alist**[**k**].**pasport **=** left**[**i**].**pasport  i **=** i **+** 1  **else:**  alist**[**k**].**pasport **=** right**[**j**].**pasport  j **=** j **+** 1  k **=** k **+** 1  **if** start **+** i **<** mid**:**  **while** k **<** end**:**  alist**[**k**].**pasport **=** left**[**i**].**pasport  i **=** i **+** 1  k **=** k **+** 1  **else:**  **while** k **<** end**:**  alist**[**k**].**pasport **=** right**[**j**].**pasport  j **=** j **+** 1  k **=** k **+** 1 |