Министерство образования Республики Беларусь

Учреждение образования

«Брестский Государственный технический университет»

Кафедра ИИТ

**Лабораторная работа №1**

По дисциплине: «Аппаратное обеспечение интеллектуальных систем»

Тема: «Применение хэширование для ассоциативного поиска по ключам»

**Выполнил:**

Студент 2 курса

Группы ИИ-21

Кирилович А. А.

**Проверил:**

Михно Е.В.

Брест 2023

**Цель работы:** изучить основные принципы хэширования, построить программную модель, обеспечивающую формирование хэш-таблиц и ассоциативный поиск по ключам.

**Задание:**

**1)** Написать на ЯВУ программу, осуществляющую просмотр, модификацию и сохранение таблицы записей. Реализовать добавление, удаление, редактирование записей и поиск по ключу перебором. Таблица должна содержать N строк (в таблицу заносится n записей) и не менее трех полей (с одним ключевым полем).

**2)** Написать на ЯВУ программу, которая преобразует таблицу данных из первого задания в хэш-таблицу на N строк (n записей). Для хэш-таблицы реализовать возможности добавления, удаления, модификации и поиска записей. Хэш-функция и метод обработки коллизий выбираются согласно варианту. Рассчитать вероятность возникновения коллизий (количество возникших коллизий при добавлении данных разделить на количество добавленных в хэш-таблицу записей) и среднее количество обращений к таблице во время поиска записи по ключу (сумма количества обращений к таблице при поиске каждой записи в хэш-таблице деленная на количество записей в хэш-таблице).

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вариант** | **Тип хэш-функции** | **Метод обработки коллизий** | **Количест-во записей n** | **Объем хэш-таблицы N** |
| 4. | Мультипликативный (все символы) 2 цифры + метод квадрата | линейное зондирование | 40 | 64 |

Код для задания 1:

import random as rn

import string

import csv

def rn\_str():

    return ''.join(rn.choices(string.ascii\_letters, k=rn.randint(5, 11)))

class Table:

    def \_\_init\_\_(self, key\_field, fields):

        self.key\_field = key\_field

        self.fields = fields

        self.table = []

    def add\_record(self, values):

        self.table.append(values)

    def remove\_record(self, key\_value):

        for i in range(len(self.table)):

            if self.table[i][0] == key\_value:

                self.table.pop(i)

                break

    def update\_record(self, key\_value, new\_values):

        for i in range(len(self.table)):

            if self.table[i][0] == key\_value:

                self.table[i] = new\_values

                break

    def search\_record(self, key\_value):

        for record in self.table:

            if record[0] == key\_value:

                return record

        return None

    def write\_to\_csv(self, filename):

        with open(filename, 'w', newline='') as csvfile:

            writer = csv.writer(csvfile)

            writer.writerow([self.key\_field] + self.fields)

            for record in self.table:

                writer.writerow(record)

    def from\_csv(self, filename):

        with open(filename, 'r', newline='') as csvfile:

            reader = csv.reader(csvfile)

            self.key\_field, \*self.fields = next(reader)

            self.table = []

            for row in reader:

                self.table.append(row)

    def create\_records(self, n, numerical=True):

        for i in range(n):

            self.add\_record([str(i + 1) if numerical else rn\_str(),

                 \*[rn\_str() for \_ in range(len(self.fields))]])

if \_\_name\_\_ == '\_\_main\_\_':

    n = 40

    table = Table('key', ['first', 'second', 'third'])

    table.create\_records(n, False)

    table.write\_to\_csv('table.csv')

Исходная таблица:

|  |  |  |  |
| --- | --- | --- | --- |
| key | first | second | third |
| wojxbHJmZbe | jfIwXpjnXZs | WaFUnwOj | bMDMz |
| CeBIQ | VNmJZl | UQpaF | RtgWnkJbbEc |
| wlNygpOzYGk | LiNHUSyK | psoum | qpSwkZ |
| kxIxqi | gCGiKfUucn | mAnpkZeFSSO | dSZXfEAjj |
| YoJXtlo | WLERWsl | mXIFHDTcYlt | FUWHuUK |
| TEjEwxJH | wDBeefwgW | WKDDLZv | mczGqGj |
| XLFZNs | KtGGc | wuPyKPgp | nDMBXXi |
| JcwmN | qxwCXtdfd | wUsRN | hfQvLgFlh |
| LxGDqqZ | IODqs | fKCmp | HUpTgDMJKx |
| SQtZs | jKfxpOESxEp | Wnegx | eLvXn |
| qizNkkVfr | UdLWR | koizixHr | zWQHOGnNc |
| fZsKaE | SIlfDtYZ | nksNrIkYF | apzcViBkEGF |
| xKAnRnFoOQ | JnxDKM | yoRDHK | KXXSXcXLF |
| lJzSrVIB | GXdkIhvBrs | uweIYJQNRf | hdpJPMRR |
| dsbSh | eGAlHZbF | TjXuXagexq | AFObMJ |
| VumGeO | bZOnaRXxC | XONqWKnL | SHYGyqruEk |
| THuAcu | rKbAQQGTlP | idPmoTZNnAm | ctzcYHit |
| FdtcI | XWJBs | pOQVLFrMQE | qJHHJz |
| DxRdJWNmd | MHJPk | sTjpqYzN | QJhJhrwMr |
| BMohi | RHdbQxMms | HkXfxdtxyN | NyZEnxrvQL |
| WalRyfaDwLi | bVFYQ | NXEKsXdRR | PRlki |
| MoltrWJWjR | xVZuQmqh | DKAroRs | PIZfzfLz |
| lCnSVkWvYQH | Wdzglgqts | JVZqXZJSl | VihzT |
| yciWk | LIPZVIRk | IkgfjmHbYDZ | AaCgMixIj |
| Oopqv | RCZhsRTej | PYwVAqYMIf | rvJYBHL |
| drwfmRmo | ReQMZYUmi | EgQtiwKetRW | YjbfE |
| PuvONTt | ZMiqDmbYZE | lPCfRV | COvFkl |
| cbCuGXCvcZ | pWUTtagbbDY | gEmpKXw | MgQeLSR |
| gFlCiIiBVcB | sCDtgfS | WdIyDONG | pSrVjXZrH |
| WYZFdJ | MyDyKPcFsFq | pSMmpbGH | GEiraZ |
| faAONvE | JUvuZGD | IsslOpFUMv | mYOGVqKXHl |
| qkrPudu | mvPQK | JFAWQ | vypQqjoCNy |
| dEKDGy | bvwWQQOoQp | NMqPbna | lYhnPfJSixN |
| hkStu | HgwILOPP | kSxttaxzz | avLLrNZnpL |
| GXPHrgN | cYaiw | WenAW | cxoFTRlMujJ |
| Zedaoy | FvqlnzBybS | woRGivalD | QCDDdMFOTB |
| ySedKZkMn | ePCRfWhvD | WFRphOmOKS | BQSvPDi |
| CSdBVj | doFliFEi | bzvhdbhmyBZ | BHJYMNZYWek |
| BqmXbT | CxFZSFFijd | QsoSleClYxx | AtnxgwPU |
| TFETlUXW | hTofjPwq | rHOut | urVRY |

Код для взаимодействия:

from task1 import Table

def menu():

    key\_field = input('Enter the name of the key field: ')

    fields = input('Enter the names of the other fields (comma-separated): ').split(',')

    table = Table(key\_field, fields)

    while True:

        print("\n=== Main Menu ===")

        print("1. Add Record")

        print("2. Remove Record")

        print("3. Update Record")

        print("4. Search Record")

        print("5. Show Table")

        print("6. Exit")

        choice = input("Enter your choice: ")

        if choice == "1":

            values = []

            values.append(input(f"Enter value for {key\_field}: "))

            for field in fields:

                values.append(input(f"Enter value for {field}: "))

            table.add\_record(values)

        elif choice == "2":

            key\_value = input(f"Enter {key\_field} of record to remove: ")

            table.remove\_record(key\_value)

        elif choice == "3":

            key\_value = input(f"Enter {key\_field} of record to update: ")

            record = table.search\_record(key\_value)

            if record:

                print(f"Current values for record {key\_value}:")

                print(f"{key\_field}: {record[0]}")

                for i, field in enumerate(fields):

                    print(f"{field}: {record[i+1]}")

                new\_values = []

                new\_values.append(key\_value)

                for field in fields:

                    new\_values.append(input(f"Enter new value for {field}: "))

                table.update\_record(key\_value, new\_values)

        elif choice == "4":

            key\_value = input(f"Enter {key\_field} of record to search: ")

            record = table.search\_record(key\_value)

            if record:

                print(f"Record found:")

                print(f"{key\_field}: {record[0]}")

                for i, field in enumerate(fields):

                    print(f"{field}: {record[i+1]}")

            else:

                print("Record not found.")

        elif choice == "5":

            print([key\_field, \*fields])

            for record in table.table:

                print(record)

        elif choice == "6":

            print("Goodbye!")

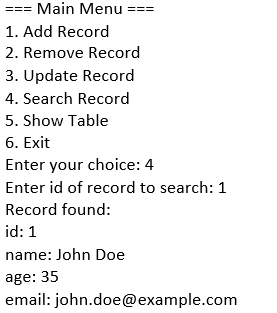
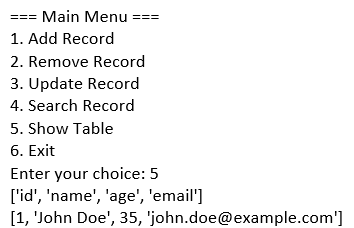
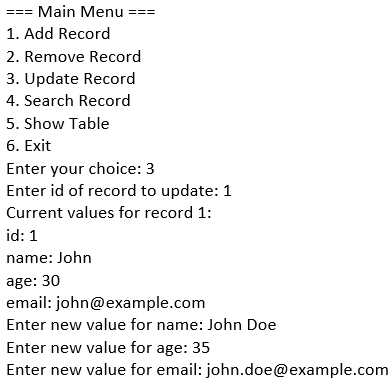
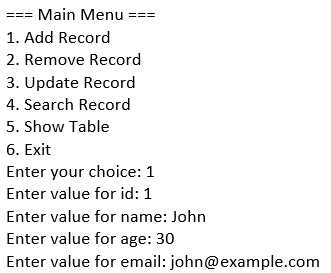
            break

        else:

            print("Invalid choice. Try again.")

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

    menu()



Код для задания 2:

import csv

from task1 import rn\_str

class HashTable:

    def \_\_init\_\_(self, size, key\_field, fields):

        self.size = size

        self.key\_field = key\_field

        self.fields = fields

        self.table = [None] \* size

        self.collision\_count = 0

        self.search\_count = 0

    def hash\_function(self, key, h=None):

        if h is None:

            h = 1

            for char in key:

                h \*= ord(char)

        h = str(h \*\* 2)

        middle = len(h) // 2

        h = int(h[middle - 1:middle + 2]) % self.size

        i = 0

        while i < self.size:

            if self.table[h] is None or self.table[h][0] == key:

                break

            else:

                if len(self.table[h]) == len(self.fields) + 1:

                    self.table[h].append(1)

                elif len(self.table[h]) == len(self.fields) + 2:

                    self.table[h][-1] += 1

                self.collision\_count += 1

                i += 1

                h = (h + i)

        return h % self.size

    def add\_record(self, values):

        key\_value = values[0]

        h = self.hash\_function(key\_value)

        self.table[h] = values

    def remove\_record(self, key\_value):

        h = self.hash\_function(key\_value)

        self.table[h] = None

    def update\_record(self, key\_value, new\_values):

        h = self.hash\_function(key\_value)

        self.table[h] = new\_values

    def search\_record(self, key\_value):

        h = self.hash\_function(key\_value)

        return self.table[h]

    def write\_to\_csv(self, filename):

        with open(filename, 'w', newline='') as csvfile:

            writer = csv.writer(csvfile)

            writer.writerow([self.key\_field] + self.fields)

            for record in self.table:

                if record is not None:

                    writer.writerow(record)

    def from\_csv(self, filename):

        with open(filename, 'r', newline='') as csvfile:

            reader = csv.reader(csvfile)

            self.key\_field, \*self.fields = next(reader)

            self.table = [None] \* self.size

            for row in reader:

                self.add\_record(row)

    def create\_records(self, n, numerical=True):

        for i in range(n):

            self.add\_record([str(i + 1) if numerical else rn\_str(),

                 \*[rn\_str() for \_ in range(len(self.fields))]])

    def collision\_probability(self):

        n = 0

        for i in range(len(self.table)):

            if self.table[i] is None:

                continue

            else:

                n += 1

        return self.collision\_count / n

    def average\_search\_count(self):

        total\_search\_count = 0

        for record in self.table:

            if record is not None:

                key\_value = record[0]

                self.search\_record(key\_value)

                total\_search\_count += self.search\_count

                self.search\_count = 0

        return total\_search\_count / len(self.table)

if \_\_name\_\_ == '\_\_main\_\_':

    N = 64

    hash\_table = HashTable(N, 'key', ['first', 'second', 'third'])

    hash\_table.from\_csv('table.csv')

    hash\_table.write\_to\_csv('hash\_table.csv')

    print(f"collision\_probability: {hash\_table.collision\_probability()}")

    print(f"average\_search\_count: {hash\_table.average\_search\_count()}")

Хеш-таблица:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| key | first | second | third |  |
| kxIxqi | gCGiKfUucn | mAnpkZeFSSO | dSZXfEAjj | 1 |
| xKAnRnFoOQ | JnxDKM | yoRDHK | KXXSXcXLF | 1 |
| CeBIQ | VNmJZl | UQpaF | RtgWnkJbbEc | |
| lJzSrVIB | GXdkIhvBrs | uweIYJQNRf | hdpJPMRR | |
| gFlCiIiBVcB | sCDtgfS | WdIyDONG | pSrVjXZrH |  |
| LxGDqqZ | IODqs | fKCmp | HUpTgDMJKx | |
| DxRdJWNmd | MHJPk | sTjpqYzN | QJhJhrwMr | |
| SQtZs | jKfxpOESxEp | Wnegx | eLvXn |  |
| yciWk | LIPZVIRk | IkgfjmHbYDZ | AaCgMixIj |  |
| fZsKaE | SIlfDtYZ | nksNrIkYF | apzcViBkEGF | 1 |
| cbCuGXCvcZ | pWUTtagbbDY | gEmpKXw | MgQeLSR |  |
| XLFZNs | KtGGc | wuPyKPgp | nDMBXXi | 1 |
| lCnSVkWvYQH | Wdzglgqts | JVZqXZJSl | VihzT |  |
| YoJXtlo | WLERWsl | mXIFHDTcYlt | FUWHuUK | 2 |
| dsbSh | eGAlHZbF | TjXuXagexq | AFObMJ | 1 |
| VumGeO | bZOnaRXxC | XONqWKnL | SHYGyqruEk | 3 |
| WYZFdJ | MyDyKPcFsFq | pSMmpbGH | GEiraZ | 1 |
| BMohi | RHdbQxMms | HkXfxdtxyN | NyZEnxrvQL | 1 |
| wojxbHJmZbe | jfIwXpjnXZs | WaFUnwOj | bMDMz |  |
| ySedKZkMn | ePCRfWhvD | WFRphOmOKS | BQSvPDi |  |
| CSdBVj | doFliFEi | bzvhdbhmyBZ | BHJYMNZYWek | |
| Oopqv | RCZhsRTej | PYwVAqYMIf | rvJYBHL | 1 |
| TEjEwxJH | wDBeefwgW | WKDDLZv | mczGqGj | 2 |
| dEKDGy | bvwWQQOoQp | NMqPbna | lYhnPfJSixN | 1 |
| wlNygpOzYGk | LiNHUSyK | psoum | qpSwkZ | 3 |
| MoltrWJWjR | xVZuQmqh | DKAroRs | PIZfzfLz |  |
| Zedaoy | FvqlnzBybS | woRGivalD | QCDDdMFOTB | |
| qkrPudu | mvPQK | JFAWQ | vypQqjoCNy | |
| THuAcu | rKbAQQGTlP | idPmoTZNnAm | ctzcYHit |  |
| TFETlUXW | hTofjPwq | rHOut | urVRY |  |
| FdtcI | XWJBs | pOQVLFrMQE | qJHHJz |  |
| qizNkkVfr | UdLWR | koizixHr | zWQHOGnNc | 1 |
| PuvONTt | ZMiqDmbYZE | lPCfRV | COvFkl |  |
| JcwmN | qxwCXtdfd | wUsRN | hfQvLgFlh |  |
| GXPHrgN | cYaiw | WenAW | cxoFTRlMujJ | |
| drwfmRmo | ReQMZYUmi | EgQtiwKetRW | YjbfE |  |
| faAONvE | JUvuZGD | IsslOpFUMv | mYOGVqKXHl | |
| WalRyfaDwLi | bVFYQ | NXEKsXdRR | PRlki |  |
| hkStu | HgwILOPP | kSxttaxzz | avLLrNZnpL | |
| BqmXbT | CxFZSFFijd | QsoSleClYxx | AtnxgwPU | |

Код для взаимодействия:

from task2 import HashTable

def menu():

    size = int(input('Enter the size of the hash table: '))

    key\_field = input('Enter the name of the key field: ')

    fields = input('Enter the names of the other fields (comma-separated): ').split(',')

    hash\_table = HashTable(size, key\_field, fields)

    while True:

        print('\nHash Table Operations:')

        print('1. Add Record')

        print('2. Remove Record')

        print('3. Update Record')

        print('4. Search Record')

        print('5. Show Table')

        print('6. Exit')

        choice = int(input('Enter your choice: '))

        if choice == 1:

            values = input(f'Enter the values for {key\_field} and {",".join(fields)}: ').split(',')

            hash\_table.add\_record(values)

        elif choice == 2:

            key\_value = input(f'Enter the value of the {key\_field} field to remove the record: ')

            hash\_table.remove\_record(key\_value)

        elif choice == 3:

            key\_value = input(f'Enter the value of the {key\_field} field to update the record: ')

            new\_values = input(f'Enter the new values for {key\_field} and {",".join(fields)}: ').split(',')

            hash\_table.update\_record(key\_value, new\_values)

        elif choice == 4:

            key\_value = input(f'Enter the value of the {key\_field} field to search for the record: ')

            record = hash\_table.search\_record(key\_value)

            if record is None:

                print('Record not found.')

            else:

                print(f'Record found: {record}')

        elif choice == 5:

            print([key\_field, \*fields])

            for record in hash\_table.table:

                print(record)

        elif choice == 6:

            print("Goodbye!")

            break

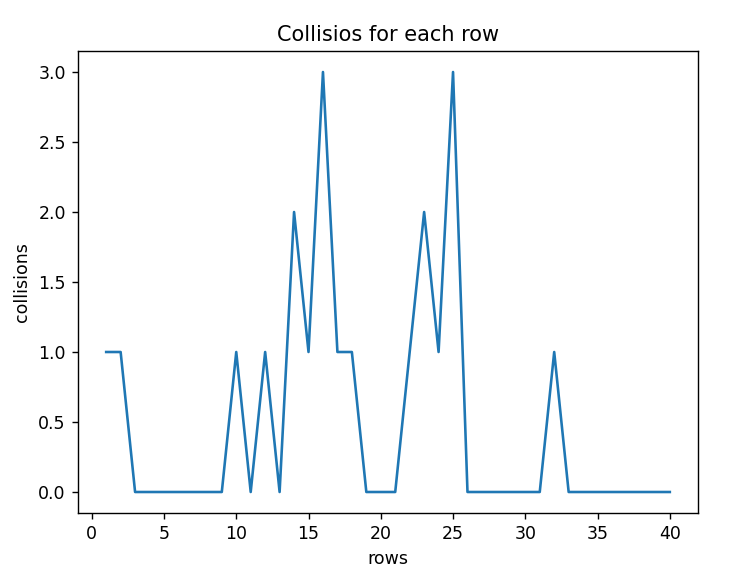
        else:

            print('Invalid choice. Please try again.')

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

    menu()

График коллизий для каждой строки в хеш-таблице:



**Вывод:** изучил основные принципы хэширования, построил программную модель, обеспечивающую формирование хэш-таблиц и ассоциативный поиск по ключам.