Министерство образования Республики Беларусь

Учреждение образования

“Брестский государственный технический университет”

Кафедра интеллектуально-информационных технологий

Лабораторная работа №1

По дисциплине «Языковые процессы интеллектуальных систем»

Выполнил:

студент 4 курса

группы ИИ-22

Копанчук Е. Р.

Проверил:

Монтик Н. С.

Брест-2024

**Цель работы:** изучить основные методы организации хеш-таблиц, получить представление о преимуществах и недостатках, присущих различным методам организации таблиц.

Для выполнения лабораторной работы требуется написать программу, которая получает на входе набор идентификаторов, организует таблицу по заданному методу и позволяет осуществить поиск идентификатора в этой таблице.

Список идентификаторов задан в виде текстового файла. Длина идентификаторов ограничена 32 символами.

**Задание**

7. Таблица строится с использованием хеш-функции на основе суммы трех первых букв идентификатора. Одинаковые элементы помещаются в одну ячейку, внутри которой организуется упорядоченный список.

**Код программы:**

# Функция для вычисления хеш-значения на основе суммы кодов первых трех символов

def hash\_function(identifier):

if len(identifier) < 3:

raise ValueError("Идентификатор должен содержать минимум 3 символа")

return sum(ord(c) for c in identifier[:3])

# Функция для построения хеш-таблицы

def build\_hash\_table(identifiers):

hash\_table = {}

for identifier in identifiers:

hash\_value = hash\_function(identifier)

if hash\_value not in hash\_table:

hash\_table[hash\_value] = []

hash\_table[hash\_value].append(identifier)

# Сортируем списки внутри каждой ячейки

for key in hash\_table:

hash\_table[key].sort()

return hash\_table

# Функция для вывода хеш-таблицы

def print\_hash\_table(hash\_table):

print("| Хеш-значение | Идентификаторы")

for hash\_value, identifiers in hash\_table.items():

#print("Хеш-значение Идентификаторы")

print(f"| {hash\_value} ", " | ", f"{', '.join(identifiers)}")

# Функция для поиска идентификатора в хеш-таблице

def search\_in\_hash\_table(hash\_table, identifier):

try:

hash\_value = hash\_function(identifier)

if hash\_value in hash\_table:

return identifier in hash\_table[hash\_value]

return False

except ValueError as e:

# Обрабатываем ситуацию с коротким идентификатором

print(f"Ошибка: {e}")

return None

# Функция для чтения идентификаторов из файла

def read\_identifiers\_from\_file(file\_path):

with open(file\_path, 'r', encoding = 'utf-8') as file:

identifiers = [line.strip() for line in file.readlines() if len(line.strip()) <= 32]

return identifiers

# Основная программа

def main():

# Чтение файла с идентификаторами

file\_path = "D:/7 семестр/ЯПИС лабы (Монтик)/ЯПИС лаба №1/Футбольные команды.txt"

identifiers = read\_identifiers\_from\_file(file\_path)

# Построение хэш-таблицы

hash\_table = build\_hash\_table(identifiers)

# Вывод хеш-таблицы

print("Хеш-таблица:")

print\_hash\_table(hash\_table)

# Цикл для многократного поиска

while True:

# Поиск идентификатора

search\_id = input("\nВведите идентификатор для поиска: ")

result = search\_in\_hash\_table(hash\_table, search\_id)

if result is None:

print("Попробуйте снова.")

elif result:

print(f"Идентификатор {search\_id} найден в хеш-таблице")

else:

print(f"Идентификатор {search\_id} не найден в хеш-таблице")

# Запрос на продолжение

continue\_search = input("\nХотите продолжить поиск? (да/нет): ").strip().lower()

# Проверка на случай пустого ввода или неверного ответа

if continue\_search not in ['да', 'нет']:

print("Неправильный ввод. Введите 'да' или 'нет'.")

elif continue\_search == 'нет':

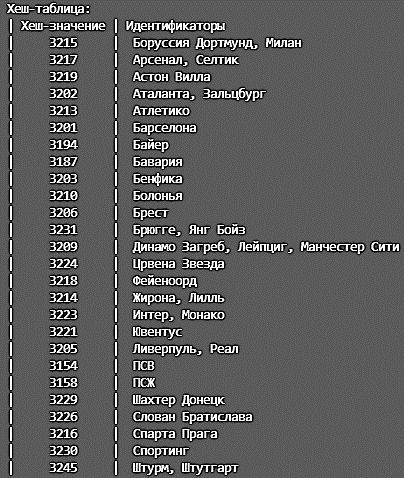
print("Завершение программы.")

break

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

main()

Результаты программы:



**Вывод:** изучил основные методы организации хеш-таблиц, получил представление о преимуществах и недостатках, присущих различным методам организации таблиц.