Белорусский государственный технологический университет

Факультет информационных технологий

Кафедра Программной Инженерии

Лабораторная работа 14

По дисциплине «Основы алгоритмизации и программирования»

На тему «Хеш-таблицы c открытой адресацией»

Выполнил:

Студент 1 курса 3 группы

Шатерник Г.И.

Преподаватель: асс. Андронова М.В.

2023, Минск

5. Использовать в проекте функции универсального хеширования и модульного. Сравнить время поиска информации.

|  |
| --- |
| Программный код  #include <stdio.h>  #include <stdlib.h>  #include <time.h>  #include<iostream>  #define TABLE\_SIZE 100 // Размер хеш-таблицы  // Структура для представления элемента данных  typedef struct {  int key;  int value;  } DataItem;  // Структура для представления хеш-таблицы  typedef struct {  DataItem\*\* array; // Массив указателей на элементы данных  } HashTable;  // Функция хеширования с использованием универсального хеширования  int universalHash(int key) {  // Здесь вы можете реализовать свою собственную функцию универсального хеширования  // Возвращаемое значение - хеш-код для ключа  // Для простоты примера, я просто использую остаток от деления на размер хеш-таблицы  return key % TABLE\_SIZE;  }  // Функция хеширования с использованием модульного хеширования  int moduloHash(int key) {  // Здесь вы можете реализовать свою собственную функцию модульного хеширования  // Возвращаемое значение - хеш-код для ключа  // Для простоты примера, я просто использую остаток от деления на размер хеш-таблицы  return key % TABLE\_SIZE;  }  // Функция создания хеш-таблицы  HashTable\* createHashTable() {  HashTable\* table = (HashTable\*)malloc(sizeof(HashTable));  table->array = (DataItem\*\*)calloc(TABLE\_SIZE, sizeof(DataItem\*));  return table;  }  // Функция для вставки элемента в хеш-таблицу  void insert(HashTable\* table, int key, int value) {  DataItem\* item = (DataItem\*)malloc(sizeof(DataItem));  item->key = key;  item->value = value;  int hash = universalHash(key); // Вычисляем хеш-код для ключа  while (table->array[hash] != NULL) {  hash = (hash + 1) % TABLE\_SIZE; // Применяем метод линейного пробирования  }  table->array[hash] = item;  }  // Функция для поиска значения по ключу в хеш-таблице с использованием универсального хеширования  int searchUniversal(HashTable\* table, int key) {  int hash = universalHash(key); // Вычисляем хеш-код для ключа  while (table->array[hash] != NULL) {  if (table->array[hash]->key == key) {  return table->array[hash]->value; // Найден элемент с заданным ключом  }  hash = (hash + 1) % TABLE\_SIZE; // Применяем метод линейного пробирования  }  return -1; // Элемент с заданным ключом не найден  }  // Функция для поиска значения по ключу в хеш-таблице с использованием модульного хеширования  int searchModulo(HashTable\* table, int key) {  int hash = moduloHash(key); // Вычисляем хеш-код для ключа  while (table->array[hash] != NULL) {  if (table->array[hash]->key == key) {  return table->array[hash]->value; // Найден элемент с заданным ключом  }  hash = (hash + 1) % TABLE\_SIZE; // Применяем метод линейного пробирования  }  return -1; // Элемент с заданным ключом не найден  }  // Функция для вывода хеш-таблицы  void printHashTable(HashTable\* table) {  for (int i = 0; i < TABLE\_SIZE; i++) {  if (table->array[i] != NULL) {  printf("Хеш: %d, Ключ: %d, Значение: %d\n", i, table->array[i]->key, table->array[i]->value);  }  }  }  // Функция освобождения памяти, занимаемой хеш-таблицей  void freeHashTable(HashTable\* table) {  for (int i = 0; i < TABLE\_SIZE; i++) {  if (table->array[i] != NULL) {  free(table->array[i]); // Освобождаем память, занимаемую элементом данных  }  }  free(table->array); // Освобождаем память, занимаемую массивом указателей  free(table); // Освобождаем память, занимаемую хеш-таблицей  }  int main() {  setlocale(LC\_ALL, "Russian");  HashTable\* table = createHashTable();  // Вставка элементов в хеш-таблицу  insert(table, 1, 10);  insert(table, 2, 20);  insert(table, 3, 30);  // Измерение времени поиска с использованием универсального хеширования  clock\_t startUniversal = clock();  int value1 = searchUniversal(table, 1);  clock\_t endUniversal = clock();  double timeUniversal = (double)(endUniversal - startUniversal) / CLOCKS\_PER\_SEC;  // Измерение времени поиска с использованием модульного хеширования  clock\_t startModulo = clock();  int value2 = searchModulo(table, 1);  clock\_t endModulo = clock();  double timeModulo = (double)(endModulo - startModulo) / CLOCKS\_PER\_SEC;  // Вывод хеш-таблицы  printHashTable(table);  // Вывод результатов поиска  printf("Значение для ключа 1 с использованием универсального хеширования: %d\n", value1);  printf("Значение для ключа 1 с использованием модульного хеширования: %d\n", value2);  // Вывод времени поиска  printf("Время поиска с использованием универсального хеширования: %f сек\n", timeUniversal);  printf("Время поиска с использованием модульного хеширования: %f сек\n", timeModulo);  // Освобождение памяти  freeHashTable(table);  return 0;  } |
| Вывод: |