МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

**ИНТЕЛЕКТУАЛЬНЫЕ ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ**

ОТЧЁТ

По лабораторной работе № \_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Выполнил:

Студент группы ИИ-22

Копанчук Евгений Романович

Проверил:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Брест – 2023

**Ход работы**

**Задание: МЕНЮ:**

|  |  |
| --- | --- |
| **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). |  |

**ТЕСТ:**

|  |  |
| --- | --- |
|  |  |
|  |  |

**КЛАССЫ:**

|  |  |
| --- | --- |
| class HashTable : public BaseTable  {  private:  unsigned collisionCounter\_;  unsigned allValCounter\_;  unsigned m\_;  std::vector<bool> collision\_mask\_;  public:  HashTable()  HashTable(int const& size, int const& columns);  HashTable(const std::vector<std::vector<std::string>>& values);  HashTable(const HashTable& obj);  ~HashTable();  unsigned long long hash\_XOR(const std::string& value);  unsigned long long hash\_Div(const std::string& value);  bool insert(const std::vector<std::string>& value) override;  bool remove(const std::string& value) override;  int search(const std::string& value) override;  bool replace(const std::vector<std::string>& old\_value,  const std::vector<std::string>& new\_value) override;  float collision\_info();  std::vector<bool> collision\_mask();  void draw\_collision();  }; | class Table : public BaseTable  {  public:  Table();  Table(int const& size, int const& columns);  Table(const std::vector<std::vector<std::string>>& values);  bool insert(const std::vector<std::string>& value) override;  bool remove(const std::string& value) override;  int search(const std::string& value) override;  bool replace(const std::vector<std::string>& old\_value,  const std::vector<std::string>& new\_value) override;  }; |