МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ

“БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ”

КАФЕДРА ИИТ

ОТЧЁТ

по лабораторной работе №4

**«Бинарные деревья поиска и хеш-таблицы»**

Выполнил:

Студент 1 курса

группы ПО-9

Мисиюк Алексей Сергеевич

Проверила:

Войцехович О. Ю.

Брест 2022

**Цель работы:** провести экспериментальное исследование эффективности бинарных деревьев поиска и хеш-таблиц.

**Постановка задачи**

Требуется реализовать на языке C две библиотеки для работы с бинарным деревом поиска (Binary search tree) и хеш-таблицей (Hash table). Ключом в обоих случаях является строка (char []), а значением целое число (int).

Функции для работы с бинарным деревом поиска должны быть помещены в файлы bstree.cpp (реализация функций) и bstree.h (объявление функций). В файлах необходимо реализовать следующие функции:

- struct bstree \*bstree\_create(char \*key, int value)

- void bstree\_add(struct bstree \*tree, char \*key, int value)

- struct bstree \*bstree\_lookup(struct bstree \*tree, char \*key)

- struct bstree \*bstree\_min(struct bstree \*tree)

- struct bstree \*bstree\_max(struct bstree \*tree)

Функции для работы с хеш-таблицей должны быть помещены в файлы hashtab.cpp (реализация функций) и hashtab.h (объявление функций). В файлах необходимо реализовать следующие функции:

- unsigned int hashtab\_hash(char \*key) /\* см лекции \*/

- void hashtab\_init(struct listnode \*\*hashtab)

- void hashtab\_add(struct listnode \*\*hashtab, char \*key, int value)

- struct listnode \*hashtab\_lookup(struct listnode \*\*hashtab, char \*key)

- void hashtab\_delete(struct listnode \*\*hashtab, char \*key)

**Порядок выполнения работы**

*Эксперимент 1 Сравнение эффективности поиска элементов в бинарном*

*дереве поиска и хеш-таблице в среднем случае (average case)*

|  |  |  |  |
| --- | --- | --- | --- |
| # | Кол-во элементов | Время выполнения bstree\_lookup, c | Время выполнения hashtab\_lookup, c |
| 1 | 10000 | 0 | 0 |
| 2 | 20000 | 0 | 0 |
| 3 | 30000 | 0 | 0 |
| 4 | 40000 | 0 | 0 |
| 5 | 50000 | 0 | 0 |
| 6 | 60000 | 0 | 0 |
| 7 | 70000 | 0 | 0 |
| 8 | 80000 | 0 | 0 |
| 9 | 90000 | 0 | 0 |
| 10 | 100000 | 0 | 0 |
| 11 | 110000 | 0 | 0 |
| 12 | 120000 | 0 | 0 |
| 13 | 130000 | 0 | 0 |
| 14 | 140000 | 0 | 0 |
| 15 | 150000 | 0 | 0 |
| 16 | 160000 | 0 | 0 |
| 17 | 170000 | 0 | 0 |
| 18 | 180000 | 0 | 0 |
| 19 | 190000 | 0 | 0 |
| 20 | 200000 | 0 | 0 |

*Эксперимент 2 Сравнение эффективности добавления элементов в бинарное*

*дерево поиска и хеш-таблицу*

*Эксперимент 3 Сравнение эффективности поиска элементов в бинарном*

*дереве поиска и хеш-таблице в худшем случае (worst case)*

*Эксперимент 4 Исследование эффективности поиска минимального элемента*

*в бинарном дереве поиска в худшем и среднем случаях*

*Эксперимент 5 Исследование эффективности поиска максимального*

*элемента в бинарном дереве поиска в худшем и среднем случаях*

**Вывод:** в результате исследования .