Лабораторная работа №6

«Программирование с использованием деревьев. Хэширование»

Задание 1.
Исходная информация в виде массива находится в компоненте StringGrid или его аналоге. Каждый элемент массива содержит строку текста и целочисленный ключ (например, Ф.И.О. и номер паспорта).

Разработать класс для работы с деревом поиска, содержащий следующие методы:

- внести информацию из массива в дерево поиска;
- сбалансировать дерево поиска;
- добавить в дерево поиска новую запись;
- по заданному ключу найти информацию в дереве поиска и отобразить ее;
- удалить из дерева поиска информацию с заданным ключом;
- распечатать информацию прямым, обратным обходом и в порядке возрастания ключа.

На основе родительского класса создать производный класс для решения задачи выбранного варианта.

Написать программу, иллюстрирующую все методы работы с деревом поиска. Результат формирования и преобразования дерева отображать в компонентах TreeView, Memo или аналогах. Написать обработчик события, реализующий работу с методом решения своего варианта.

Индивидуальные задания

1. Поменять местами информацию, содержащую максимальный и минимальный ключи.
2. Подсчитать число листьев в дереве. (Лист – это узел, из которого нет ссылок на другие узлы дерева.)
3. Удалить из дерева ветвь с вершиной, имеющей заданный ключ.
4. Определить максимальную глубину дерева, т.е. число узлов в самом длинном пути от корня дерева до листьев.
5. Определить число узлов на каждом уровне дерева.
6. Удалить из левой ветви дерева узел с максимальным значением ключа и все связанные с ним узлы.
7. Определить количество символов во всех строках, находящихся в узлах дерева.
8. Определить число листьев на каждом уровне дерева.
9. Определить число узлов в дереве, в которых есть указатель только на одну ветвь.
10. Определить число узлов в дереве, у которых есть две дочери.
11. Определить количество записей в дереве, начинающихся с определенной буквы (например а).

12. Найти среднее значение всех ключей дерева и найти узел, имеющий ближайший к этому значению ключ.
13. Найти запись с ключом, ближайшим к среднему значению между максимальным и минимальным значениями ключей.
14. Определить количество узлов в левой ветви дерева.
15. Определить количество узлов в правой ветви дерева.

Задание 2.

В соответствии со своим вариантом построить ***хеш-таблицы*** с ***открытой*** адресацией размерами 16, 64, 128, 2048 и заполнить с коллизиями. В таблице **h'(key)** – значение хеш-функции, приведшее к коллизии.

Исследовать время поиска в хеш-таблицах. Реализовать оконное приложение, в котором строятся соответствующие графики.

Индивидуальные варианты:

1. Изменить функцию вычисления хеш для решения коллизии на квадратичную функцию, которая строится на основе формулы: **$h(key, i) = (h'(key) + c_1*i + c_2*i^2) \bmod hashTableSize$**.
2. Использовать в проекте функцию универсального хеширования.
3. Изменить функцию вычисления хеш на мультипликативную функцию, которая строится на основе формулы: **$H(key) = [hashTableSize(key \cdot A \bmod 1)]$**, где **$A = (sqrt(5) - 1) / 2 = 0{,}6180339887499$**.
4. Использовать в проекте функции универсального хеширования и модульного. Сравнить количество коллизий при введении одинаковых ключей.
5. Использовать в проекте линейный алгоритм вычисления последовательности испробованных мест.
6. Использовать в проекте функции мультипликативного и модульного хеширования. Сравнить время поиска информации.
7. Изменить функцию вычисления хеш на универсальную.
8. Изменить проект с тем, чтобы использовался аддитивный метод хеширования (ключи должны быть строковыми данными).
9. Изменить хеш-функцию в проекте на модульную.
10. Изменить функцию вычисления хеш для решения коллизии на линейную функцию, которая строится на основе формулы: **$h(key, i) = (h'(key) + i) \cdot \bmod hashTableSize$**.
11. Реализовать хеш-таблицу с открытой адресацией для хранения строк. Таблица должна увеличивать свой размер вдвое при достижении 80% заполнения.

12. Реализовать динамическую хеш-таблицу с открытой адресацией для хранения строк. Таблица должна увеличивать свой размер вдвое при достижении 50% заполнения.
13. Использовать в проекте функции мультипликативного и модульного хеширования. Сравнить количество коллизий при введении одинаковых ключей.
14. Использовать в проекте функции универсального хеширования и модульного. Сравнить время поиска информации.
15. Изменить функцию вычисления хеш для решения коллизии на функцию, которая строится на основе формулы: **h(key, i) = (h1(key) + i·h2(key))·mod hashTableSize**, где **h1 (key) = key·mod hashTableSize** , **h2(key) = 1 + (key·mod hashTableSize)**.
16. Реализовать хеш-таблицу с открытой адресацией для хранения строк. Таблица должна увеличивать свой размер втрое при достижении 70% заполнения.

Задание 3.

Разработать приложение, в котором содержатся следующие классы:

Родительский класс, реализующий методы работы с хеш-таблицей на основе массива стеков **(не использовать шаблоны STL и boost)**.

Производный класс, созданный на базе родительского и реализующий метод решения своего варианта.

В приложении продемонстрировать работу всех методов работы с хеш-таблицей. Результат формирования и преобразования хеш-таблицы показывать в компоненте Memo методом Print(Memo) или его аналогах в виде строк, отображающих стеки.

Написать обработчик события, реализующий вызов метода решения своего варианта.

Индивидуальные задания

1. Создать хеш-таблицу со случайными целыми ключами в диапазоне от–50 до +50 и преобразовать ее в две таблицы. Первая должна содержать только положительные ключи, а вторая – отрицательные.
2. Создать хеш-таблицу со случайными целыми ключами и удалить из него записи с четными ключами.
3. Создать хеш-таблицу со случайными целыми ключами в диапазоне от –10 до 10 и удалить из него записи с отрицательными ключами.
4. Создать хеш-таблицу со случайными целыми ключами и найти запись с минимальным ключом.

5. Создать хеш-таблицу со случайными целыми ключами и найти запись с максимальным ключом.
6. Подсчитать, сколько элементов хеш-таблицы со случайными ключами превышает среднее значение от всех ключей.
7. Создать хеш-таблицу из случайных целых чисел и найти в ней номер стека, содержащего минимальное значение ключа.
8. Создать хеш-таблицу из случайных целых чисел и найти в ней номер стека, содержащего максимальное значение ключа.