Лабораторная работа № 4б «Работа с самобалансирующимися деревьями поиска»

Введение

Задачи

В процессе выполнения лабораторной работы необходимо решить следующие задачи:

1. Спроектировать и разработать на языке C:
   (a) Библиотеку, предоставляющую функциональность по работе с деревом поиска, в соответствии с условиями индивидуального задания.
   (b) Прикладную программу, позволяющую пользователю в диалоговом режиме работать с деревом.
2. Выполнить таймирование (или профилирование) программы. Построить графики зависимости времени выполнения операций, предусмотренных индивидуальным заданием, от количества элементов в дереве.
3. Оценить сложность реализованных алгоритмов.

Дополнительные задачи

Существует ряд дополнительных задач, не обязательных к выполнению, но позволяющих получить дополнительные баллы:

1. * Реализовать графический вывод дерева при помощи локальной внешней утилиты или библиотеки (например, graphviz). При этом, отображение дерева в графическом интерфейсе пользователя или генерация файла с изображением должно происходить автоматически, без выполнения действий вручную со стороны пользователя.
2. ** Доступ к элементам дерева реализовать через дополнительный кэш-буфер, реализованный в виде хеш-таблицы. Размер кэш-буфера ограничен числом N.
3. ** Написать программу по условию одного из вариантов, представленных в таблице 1. Особенности выполнения данного дополнительного задания:
   - Программа должна использовать разработанную ранее библиотеку.
   - Пользователь должен иметь возможность диалогового взаимодействия с программой: при запуске указать имя обрабатываемого файла, а затем — данные, поиск которых необходимо осуществить.
   - Программа должна позволять пользователю выполнять поиск произвольное количество раз без перезапуска.

Основные операции

В программе необходимо предусмотреть возможность проведения следующих операций над деревом, особенности реализации которых определяются индивидуальным заданием:

1. добавление нового элемента;
2. удаление элемента;
3. обход;
4. поиск элемента по ключу;
5. специальный поиск элемента.

Кроме того, должны быть реализованы следующие общие операции:

1. форматированный вывод дерева «в виде дерева»;
2. загрузка дерева из текстового файла следующего формата:
    - Key1
    - Info1
    - Key2
    - Info2
    - ...

Примечания

1. Логически законченные части алгоритма решения задачи должны быть оформлены в виде отдельных функций с параметрами. Использование глобальных переменных не допускается.
2. Функции для работы с деревом должны быть организованы в виде отдельной библиотеки, которая используется основной программой.
3. Функции для работы с деревом не должны быть диалоговыми, т. е. они должны принимать все необходимые данные в качестве параметров и возвращать результат работы в виде соответствующих структур данных и кодов ошибок (исключение: функции вывода дерева).
4. Диалоговые функции должны использовать описанные выше функции.
5. Программа должна осуществлять проверку корректности вводимых данных и, в случае ошибок, выдавать соответствующие сообщения, после чего продолжать работу.
6. В случае возникновения ошибочных ситуаций при выполнении операций с деревом программа должна выводить соответствующие сообщения, после чего продолжать работу.
7. Для сборки программы и библиотек должна использоваться система сборки (например: Make или CMake).
8. Библиотеки и прикладная программа должны собираться независимо друг от друга.
9. Программа должна корректным образом работать с памятью, для проверки необходимо использовать соответствующие программные средства, например: valgrind, санитайзеры, встроенные в IDE средства и т.д.

Таблица 1: Варианты дополнительного задания № 3

Типы даных		Дублирование ключей	Формулировка	Возвращаемое значение
Ключ	Значение			
Строка	Число	Разрешено	Быстрый поиска слова в текстовом файле	Вектор, состоящий из номеров строк, в которых присутствует искомое слово
		Запрещено	Быстрый поиска слова в текстовом файле	Номер строки первого вхождения искомого слова
	Строка	Разрешено	Быстрый поиск слова в текстовом файле	Вектор, состоящий из строк с расположением искомого слова в формате `<имя_файла>: <номер_строки>: <смещение_в_строке>`
		Запрещено	Быстрый поиск слова в текстовом файле	Строка формата `<имя_файла>: <номер_строки>: <смещение_в_строке>`, указывающая на первое вхождение искомого слова
Число	Число	Разрешено	Быстрый поиск числа в текстовом файле, строки которого содержат ноль или более чисел, разделённых запятой	Вектор, состоящий из номеров строк, в которых присутствует искомое число
		Запрещено	Быстрый поиск числа в текстовом файле, строки которого содержат ноль или более чисел, разделённых запятой	Номер строки первого вхождения искомого числа
	Строка	Разрешено	Быстрый поиск числа в текстовом файле, строки которого содержат ноль или более чисел, разделённых запятой	Вектор, состоящий из строк с расположением искомого числа в формате `<имя_файла>: <номер_строки>: <номер_числа_в_строке>`
		Запрещено	Быстрый поиск числа в текстовом файле, строки которого содержат ноль или более чисел, разделённых запятой	Строка формата `<имя_файла>: <номер_строки>: <номер_числа_в_строке>`, указывающая на первое вхождение искомого числа