**КУРСОВАЯ РАБОТА**

**Алгоритмы работы с таблицами**

по дисциплине «**Алгоритмы и структуры данных**»

|  |  |
| --- | --- |
| Выполнил  студент гр. 5130904/30004 | Кравченко Н. В. |
| Руководитель | Шемякин И. А. |

«\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2024

# СОДЕРЖАНИЕ

[СОДЕРЖАНИЕ 2](#_Toc167779096)

[Введение. Общая постановка задачи 3](#_Toc167779097)

[Тема: Алгоритмы работы со словарями 3](#_Toc167779098)

[Основная часть 4](#_Toc167779099)

[1. Описание алгоритма решения и используемых структур данных. 4](#_Toc167779100)

[2. Анализ алгоритмов 5](#_Toc167779101)

[3. Описание программы 8](#_Toc167779102)

[4. Описание спецификации программы 8](#_Toc167779103)

[Заключение 12](#_Toc167779104)

[Источники 13](#_Toc167779105)

# Введение. Общая постановка задачи

## Тема: Алгоритмы работы со словарями

1. Для разрабатываемого словаря реализовать основные операции:
   * INSERT (ключ, значение) – добавить запись с указанным ключом и значением
   * SEARCH (ключ) – найти запись с указанным ключом
   * DELETE (ключ) – удалить запись с указанным ключом
2. Предусмотреть обработку и инициализацию исключительных ситуаций, связанных, например, с проверкой значения полей перед инициализацией и присваиванием.

3. Программа должна быть написана в соответствии со стилем программирования: C++ Programming Style Guidelines.

4. Тесты должны учитывать, как допустимые, так и не допустимые последовательности входных данных.

**Вариант 1.3.3. Частотный словарь. Красно-черное дерево**

Разработать и реализовать алгоритм формирования частотного словаря:

* определить понятие слово
* прочитать текст и сформировать набор слов вместе с информацией о частоте их встречаемости
* определить три чаще всего встречающихся слова

Для реализации задания использовать красно-черное дерево.

# Основная часть

## Описание алгоритма решения и используемых структур данных.

Алгоритм решения:

1. Реализовать шаблонное красно-черное дерево (далее КЧД) с необходимыми методами и итераторами
2. Реализовать основную программу-терминал, принимающую последовательность команд:
3. Словарь – КЧД, в котором ключ – слово, значение – частота;
4. Структура для хранения команд – КЧД, в котором ключ – название команды, значение – функция;
5. Логика команд и поведение при неправильных входных данных
6. Протестировать программу с различными корректными/некорректными входными данными

Используемые структуры данных:

1. RedBlackTree (Красно-черное дерево)

Интерфейс:

* 1. Специальные методы и операторы (конструктор по умолчанию, конструкторы копирования и перемещения, операторы копирующего и перемещающего присваивания, деструктор)
  2. begin(), end()
  3. cbegin(), cend()
  4. clear()
  5. empty()
  6. insert(value), insert(iterator range)
  7. erase(key), erase(iterator)
  8. at(key)
  9. operator[](key)

1. std::vector

## Анализ алгоритмов

1. **Класс RedBlackTree:**

* Вставка элемента – O(log(N));
* Поиск по ключу – O(log(N));
* Удаление элемента O(log(N));

1. **Класс std::vector:**

* Вставка в конец – O(1);

1. **Алгоритм std::sort:**

* Introsort – O(N \* log(N))

1. **Команды словаря:**

* READ – O(N \* log(N));
* INSERT – O(log(N));
* SEARCH – O(log(N));
* DELETE – O(log(N));
* CLEAR – O(N);
* MOSTCOMMON – O(N \* log(N));

**Реализация красно-черного дерева:**

*Определение*

Красно-чёрным называется бинарное поисковое дерево, у которого каждому узлу сопоставлен дополнительный атрибут — цвет и для которого выполняются следующие свойства:

1. Каждый узел промаркирован красным или чёрным цветом
2. Корень и конечные узлы (листья) дерева — чёрные
3. У красного узла родительский узел — чёрный
4. Все простые пути из любого узла x до листьев содержат одинаковое количество чёрных узлов

Черная высота вершины X – число черных вершин на пути из X в лист, не учитывая саму вершину X.

Для экономии памяти фиктивные листья можно сделать одним общим фиктивным листом (nullptr).

*Вставка элемента*

Пусть вставляемый элемент – X.

Каждый элемент вставляется вместо листа по правилам бинарного дерева и имеет красный цвет. Если «отец» элемента X черный, дальнейшие действия не требуются. Если «отец» элемента X красный, нарушается свойство 3. Рассмотрим 3 случая:

1. У узла нет «дяди».
   1. Если X – правый потомок, а «отец» – левый, делаем сначала левое вращение, которое сделает «отца» левым потомком узла X.
   2. Делаем правое вращение «отца» и «деда» X. Перекрашиваем «отца» в черный, а «деда», который стал потомком в красный.

Если «отец» – левый, то порядок действий симметричен.

1. «Дядя» этого узла тоже красный. Перекрашиваем «отца» и «дядю» в черный цвет, а «деда» – в красный. Теперь рассматриваем «деда», так как он может нарушать свойство 3.
2. «Дядя» черный и правый.
   1. Если X – правый потомок, делаем сначала левое вращение, которое сделает «отца» левым потомком узла X.
   2. Делаем правое вращение «отца» и «деда» X. Перекрашиваем «отца» в черный, а «деда», который стал потомком в красный.

Если «дядя» – левый, то порядок действий симметричен.

*Удаление элемента*

Пусть удаляемый элемент – X.

При удалении элемента могут возникнуть 3 случая, в зависимости от количества наследников:

1. У X нет наследников, то удаляем узел, меняя указатель на него у родителя на nullptr.
2. Если у X один наследник, то удаляем узел, меняя указатель на него у родителя на наследника X.
3. Если у X оба наследника, то находим вершину со следующим значением ключа. У такой вершины нет левого наследника. Удаляем уже эту вершину, рассматривая случай 1 или 2, скопировав её ключ в X.

Удаляем вершину:

1. Удаление красной. При удалении красной вершины свойства не нарушаются. Восстановление не требуется.
2. Удаление черной с потомком. Так как единственным потомком черной вершиной может быть красная, то просто заносим в черную вершину данные красной и удаляем. Восстановление свойств не требуется.
3. Удаление черной вершины без потомков. Удаляем вершину, а лист на месте делаем «дважды черным» и назовем X. Требуется восстановление свойства 4.
   1. Если вершина X – корень, оставляем его просто черным.
   2. Если «брат» вершины X – красный, то делаем вращение вокруг ребра между «отцом» и «братом». Красим нового «деда» в черный, а «отца» в красный.
   3. Если «брат» X – черный, и оба его наследника – черный, то красим брата в красный. Если отец был черным, теперь рассматриваем его, как X.
   4. Если «брат» X – черный, и его левый наследник – красный, то делаем правое вращение по ребру наследника, красим «брата» в красный, а наследника в черный. Так нынешний правый наследник станет красным.
   5. Если «брат» X – черный, и его правый наследник – красный, то делаем левое вращение по ребру «брата», красим его в цвет «отца», а «отца» в черный.

## Описание программы

Программа представляет собой консольный интерфейс для работы с частотным словарем. Ввод каждой команды осуществляется на одной строке. Команды вводятся до тех пор, пока не введен EOF (ctrl+Z), или не произошла какая-либо внутренняя ошибка.

Определение слова в словаре – это последовательность или подпоследовательность символов в строке, удовлетворяющая регулярному выражению:

([A-Za-z][A-Za-z\-']\*[A-Za-z])|[IiAa]

При введении параметра командной строки «--test» программа запустится в режиме автоматического тестирования Красно-черного дерева и системы команд.

Список доступных команд:

1. READ <text\_file> – чтение текста из файла <text\_file>, и составление частотного словаря по этому тексту.
2. INSERT <word> <count> – вставка в словарь слова <word> с указанной частотной характеристикой <count>.
3. SEARCH <word> – поиск слова <word> в словаре и вывод его частотной характеристики.
4. DELETE <word> – удаление слова <word> из словаря.
5. CLEAR – полное удаление всей информации из словаря.
6. MOSTCOMMON [N] – вывод трех или N самых часто встречающихся слов в порядке убывания.

## Описание спецификации программы

Детальные требования для команд:

|  |  |  |  |
| --- | --- | --- | --- |
| Требование | Детальные требования | Данные | Ожидаемый результат |
| 1. Команда READ <text\_file> | | | |
| 1.1. текстовый файл <text\_file> читаем | 1.1. Если файла с таким названием не существует или файл невозможно прочесть, сообщение «*INVALID FILE*» | erFile.b | Сообщение: «INVALID FILE» |
| 1.2. Введенные данные верны | 1.2. Если все условия выполнены, прочитать файл, записать в словарь и ничего не вывести. | 1984.txt | Нет сообщения |
| 2. Команда INSERT <word> <count> | | | |
| 2.1. Вводимый ключ <word> является словом | 2.1. Если <word> не является словом, сообщение «*INVALID WORD*» | 2011 10 | Сообщение:  «INVALID WORD» |
| 2.2. Вводимое значение <count> является положительным числом | 2.2. Если count = 0 или не является беззнаковым числом, сообщение «*INVALID COUNT*» | sofa -12 | Сообщение:  «INVALID COUNT» |
| 2.3. Введенное слово не присутствует в словаре | 2.3. Если в словаре уже есть слово, сообщение «*WORD EXIST*» | the 200 | Сообщение:  «WORD EXIST» |
| 2.4. Введенные данные верны | 2.4. Если все условия выполнены, вставить слово с указанным количеством в словарь и ничего не вывести | brother 23 | Нет сообщения |
| 3. Команда SEARCH <word> | | | |
| 3.1. Вводимый ключ <word> является словом | 3.1. Если <word> не является словом, сообщение «*INVALID WORD*» | 0! | Сообщение:  «INVALID WORD» |
| 3.2. Введенное слово присутствует в словаре | 3.2. Если в словаре нет такого слова, сообщение «*WORD NOT FOUND*» | big | Сообщение:  «WORD NOT FOUND» |
| 3.3. Введенные данные верны | 3.3. Если все условия выполнены, вывести частоту слова | and | Сообщение:  257 |
| 4. Команда DELETE <word> | | | |
| 4.1. Вводимый ключ <word> является словом | 3.1. Если <word> не является словом, сообщение «*INVALID WORD*» | m | Сообщение:  «INVALID WORD» |
| 4.2. Введенное слово присутствует в словаре | 3.2. Если в словаре нет такого слова, сообщение «*WORD NOT FOUND*» | warning | Сообщение:  «WORD NOT FOUND» |
| 4.3. Введенные данные верны | 3.3. Если все условия выполнены, удалить слово из словаря и ничего не вывести | such | Нет сообщения |
| 5. Команда CLEAR | | | |
| 5.1. Выполняется в любом случае | 5.1. Удаляет все данные из словаря, сообщение «*DICTIONARY CLEARED*» |  | Сообщение:  «DICTIONARY CLEARED» |
| 6. Команда MOSTCOMMON [N] | | | |
| 6.1. Словарь не пустой | 6.1. Если словарь пуст, сообщение «*EMPTY DICTIONARY*» |  | Сообщение:  «EMPTY DICTIONARY» |
| 6.2. Дополнительный параметр [N] не введен | 6.2. Число выводимых слов равно трем, выполнение продолжается |  |  |
| 6.3. Вводимое значение [N] является положительным числом | 6.3. Если N = 0 или не является беззнаковым числом, сообщение «*INVALID NUMBER*» | a12 | Сообщение:  «INVALID NUMBER» |
| 6.4. Число выводимых слов больше, чем слов в словаре | 6.4. Если число слов в словаре больше 3 или N, вывести возможное число слов | 5 | Сообщение:  hey : 4  are : 1  okay : 1  you : 1 |
| 6.5. Введенные данные верны | 6.5. Если все условия выполнены, вывести заданное количество слов в порядке убывания в формате «слово : частота» | 2 | Сообщение:  the : 31542  a : 27935 |

# Заключение

В ходе курсовой работы была подробно изучена структура данных «Красно-черное дерево»: ее строение, детали реализации, сложность операций, логика итераторов для бинарного дерева. Также были изучены некоторые полезные функции стандартной библиотеки шаблонов (STL) C++.

В ходе работы были разработаны и написаны:

1. Шаблонный класс «Красно-черное дерево» с сигнатурами, похожими на std::map
2. Алгоритм обработки и диспетчеризации команд
3. Интерактивные команды для взаимодействия с пользователем
4. Тестовые данные для КЧД и системы команд

Практический итог курсовой работы – рабочее консольное приложение для взаимодействия с частотным словарем.

# Источники

* + - 1. Красно-черное дерево – Википедия – URL: [https://ru.wikipedia.org/wiki/Красно-чёрное-дерево](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B0%D1%81%D0%BD%D0%BE-%D1%87%D1%91%D1%80%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE). – (дата обращения: 18.05.2024).
      2. Красно-черное дерево – Викиконспекты – URL: [https://neerc.ifmo.ru/wiki/index.php?title=Красно-черное\_дерево](https://neerc.ifmo.ru/wiki/index.php?title=%D0%9A%D1%80%D0%B0%D1%81%D0%BD%D0%BE-%D1%87%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D0%B4%D0%B5%D1%80%D0%B5%D0%B2%D0%BE). – (дата обращения: 18.05.2024).
      3. C++ Containers library std::map – URL: <https://en.cppreference.com/w/cpp/container/map>. – (дата обращения: 18.05.2024).
      4. C++ Containers library std::vector – URL: <https://en.cppreference.com/w/cpp/container/vector>. – (дата обращения: 18.05.2024).
      5. Дистанционные занятия МФТИ – Алгоритмы и структуры данных (Красно-черные деревья), Мацкевич С. Е. 29.11.2021г. – URL: <https://youtu.be/gCPtoK07yRA?si=wVMPOfYMhrFNVJTE>. – (дата обращения: 19.05.2024).
      6. C++ Iterator Library – URL: <https://en.cppreference.com/w/cpp/iterator>. – (дата обращения: 19.05.2024).