

DDC23 Спецификация (Описание для разработчика)

Контейнер данных для использования в разработке приложений, требующих литеральные идентификаторы, типа словарей

Проектная группа	--		
Заказчик	Самостоятельный проект		
Проект	Контейнер данных для использования в разработке приложений, требующих литеральные идентификаторы, типа словарей		
Руководитель проекта	Артур Мангус		
Участники проекта	Самостоятельный проект		
Документ	4. Спецификация	Количество страниц	9
Автор документа	Артур Мангус		
Создан	15.01.2023		
Последнее изменение	15.01.2023		
Статус обработки	X	В обработке Представлен разработчикам и заказчику Одобен Закрит	

История документа

Версия	Дата	Автор изменения	Описание / замечание
0.1	16.01.2023	Артур Мангус	Редактирование обзора и общего описания
0.2	17.01.2023	Артур Мангус	Редактирование требований
0.3	18.01.2023	Артур Мангус	Разработка функциональных требований

Оглавление

1	Введение	3
1.1	Обзор содержимого спецификации	3
1.1.1	Цели	3
1.1.2	Масштаб проекта	3
2	Общее описание	3
2.1	Обзор программного продукта	4
2.2	Функции продукта	4
2.3	Характеристика пользователей	4
2.4	Общие ограничения	4
3	Требования (Основной раздел)	5
3.1	Функциональные требования	5
3.1.1	Добавление литерального идентификатора (далее «ключа»)	5
3.1.2	Поиск ключа в контейнере	6
3.1.3	Вывод данных по заданному ключу	6
3.1.4	Удаление ключа и связанных с ним данных	7
3.1.5	Редактирование данных связанных с ключом	8
3.1.6	Вывод списка всех ключей	8
3.1.7	Вывод карты ключей и ассоциированных данных	8
3.1.8	Редактирование ключа	9
3.1.9	Очистка контейнера	9
3.2	Эксплуатационные (нефункциональные) требования	9
3.2.1	Производительность	9
3.2.2	Сохранность данных	9
3.2.3	Безопасность	9
4	Модели	9

1 Введение

Цели настоящей спецификации:

- получить точную оценку стоимости, рисков и затрат времени
- основание для клиента более четко сформировать видение задания
- получение одинакового представления о продукте заказчиком и исполнителем
- выявление оптимального набора функций
- представление основы для формирования другой технической документации
- избежание дублирования задач
- структурирование проблем для простого и быстрого решения
- выявление оптимальных результатов при тестировании
- основание для создания руководства пользователя

Настоящая спецификация следует соблюдению следующих свойств:

- спецификация не должна содержать деталей реализации, в отличие от программы она указывает, что надо сделать, а не как это делать
- спецификация должна обладать формальностью (однозначностью прочтения, точностью), причем диапазон может быть очень широк – от полностью формализованного описания до слегка формализованного
- спецификация должна быть понятной (ясной, читабельной), в общем случае она должна быть более понятным описанием задачи, чем программа, так как краткость не всегда содействует ясности и понятности
- спецификация должна обладать полнотой описания задачи, ничто существенное не должно быть упущено.

1.1 Обзор содержимого спецификации

1.1.1 Цели

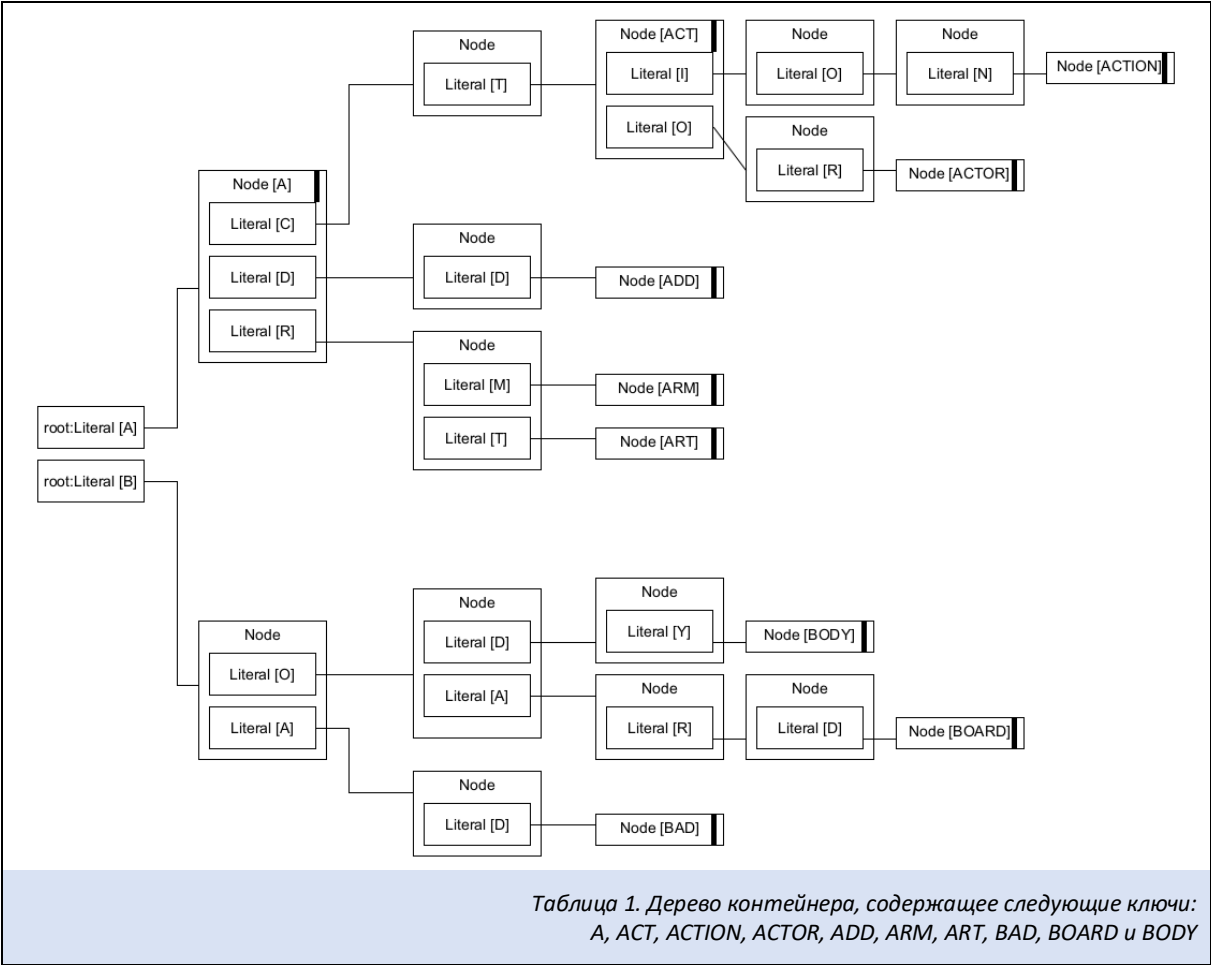
Целью проекта является разработка ассоциативного контейнера данных, который может быть использован для создания виртуальных словарей любого объема. Как и любой словарь, контейнер должен содержать уникальные ключи.

1.1.2 Масштаб проекта

Реализация контейнера должна представлять из себя одиночный заголовочный файл.

2 Общее описание

Контейнер должен представлять из себя N-арное дерево, содержащее неограниченное количество литеральных ключей любой длины, хранящее ссылки на любой тип данных. Контейнер должен обеспечивать максимальное быстродействие операций, описанных в этой спецификации. Контейнер должен принимать указатель на данные любого типа, ассоциируемые с ключами. Каждая ветвь дерева может содержать ассоциативные данные для любого узла ветви, но как минимум для последнего узла ветви данные обязательны. Если узел в ветви содержит данные, то значения от корня и до этого узла должны образовывать литеральный ключ.



2.1 Обзор программного продукта

Контейнер не должен представлять из себя самостоятельно исполняемый программный продукт. Заголовочный файл контейнера, должен интегрироваться в любой проект реализуемый на C++11 и более поздних спецификациях.

2.2 Функции продукта

Контейнер должен декларироваться с заданным типом данных. Тип данных ключа string определен по умолчанию и не должен переопределяться.

Контейнер должен реализовывать функции запросов и модификаций.

2.3 Характеристика пользователей

Интерфейсы контейнера, должны быть интуитивно понятны разработчикам, и иметь сходство с подобными интерфейсами стандартных библиотек.

2.4 Общие ограничения

Ограничения не определены.

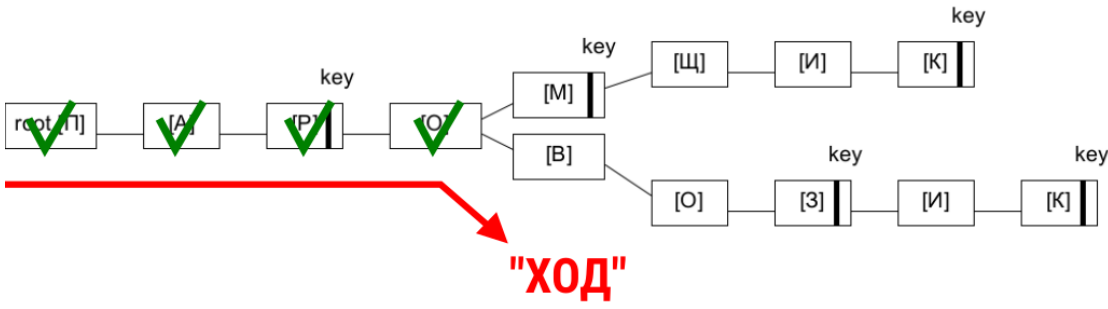
3 Требования (Основной раздел)

3.1 Функциональные требования

3.1.1 Добавление литерального идентификатора (далее «ключа»)

Функция: ADD KEY	ID: DDC23-01	Интерфейс:
<p>Входные данные: Функция должна принимать в качестве входных данных, два обязательных аргумента:</p> <ul style="list-style-type: none">• Литерал добавляемого ключа• Экземпляр данных, соответствующий задекларированному пользователем типу данных <p>Обработка данных: На основе литералов ключа, от корневого узла контейнера, создаются промежуточные взаимосвязанные узлы, образующие словарную ветвь. Конечный узел ключа содержит данные. Это означает, что в одной словарной ветви могут быть созданы множественные ключи.</p> <div><pre>graph LR root["root [П]"] --> A["[A]"] A --> P["[P]"] P -- key --> O1["[O]"] P -- key --> O2["[O]"] O1 -- key --> M["[М]"] O2 --> B["[В]"] M -- key --> Sh["[Щ]"] Sh --> I1["[И]"] I1 -- key --> K1["[К]"] B --> O3["[О]"] O3 -- key --> Z["[З]"] Z -- key --> I2["[И]"] I2 -- key --> K2["[К]"]</pre></div> <p>Так, например три ключа – «пар», «паром», «паромщик» содержаться в одной ветви словаря. Каждый узел, отмеченный флагом, должен содержать ассоциативные данные ключа. Как показано на диаграмме, одна ветвь может разделяться на множество других ветвей. Так, например два ключа «паром» и «паровоз», берут начало в одной ветви, исходящий из корневого узла, и разделяются на две следующие ветви в узле литерала «О». Соответственно узел «О» связан указателями с двумя другими узлами «М» и «В». Место деления ветви не обязательно должно содержать ключи для каждого узла литерала.</p> <p>Операции с данными: Каждый последний узел, обязательно должен содержать флаг ключа и ассоциированные с ним данные. Каждый узел, получивший флаг ключа, обязательно должен содержать ассоциативные с ним данные.</p> <p>Выходные данные: Функция должна возвращать результат булевского типа</p>		

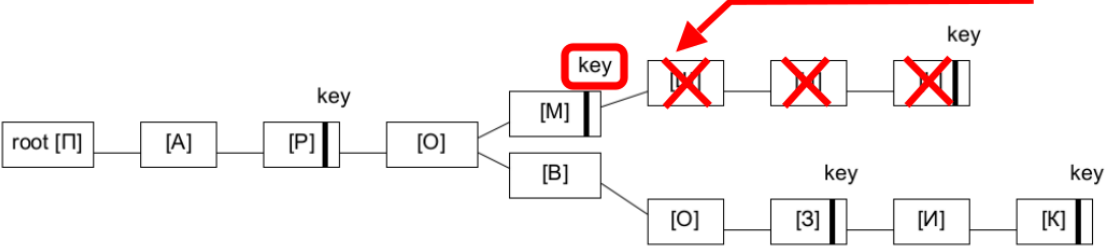
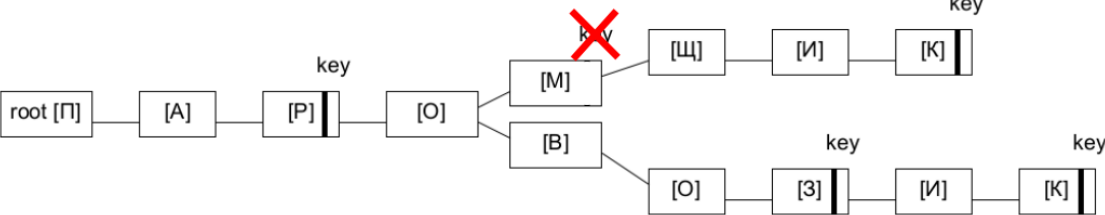
3.1.2 Поиск ключа в контейнере

Функция: SEARCH OR FIND	ID: DDC23-02	Интерфейс:
<p>Входные данные: Функция должна принимать в качестве входных данных один аргумент:</p> <ul style="list-style-type: none"> Литерал искомого ключа <p>Обработка данных: На основе вводимых данных должен производиться обход ветвей контейнера от корня к вершинам. Если при обходе ветви достигнут узел, соответствующий последнему литералу введенного запроса, и этот узел имеет установленный флаг ключа и ассоциативные данные, то функция должна вернуть положительный результат. Если достигнутый узел существует и соответствует последнему литералу введенного запроса, но флаг ключа не установлен, функция должна вернуть негативный результат. Если достигнут некоторый узел ветви, который не содержит связанных узлов со следующим в очереди литералом из введенного запроса, то функция также должна вернуть негативный результат. Так в качестве примера на диаграмме взят ключ «пароход».</p>  <p>Выходные данные: Функция должна возвращать результат булевского типа</p>		

3.1.3 Вывод данных по заданному ключу

Функция: GET DATA	ID: DDC23-03	Интерфейс:
<p>Входные данные: Функция должна принимать в качестве входных данных один аргумент:</p> <ul style="list-style-type: none"> Литерал ключа для запроса ассоциативных данных <p>Обработка данных: Функция должна выполнять обход ветвей контейнера также, как и в функции поиска ключа DDC23-02.</p> <p>Операции с данными: Однако, в качестве результата функция должна вернуть ассоциированные с ключом данные. Если ключ отсутствует, то функция должна вернуть нулевые данные декларированного типа. Так как тип данных декларируется динамически, то и контроль за возвращаемым результатом на уровне контейнера не представляется возможным, и должен производиться вне контейнера.</p> <p>Выходные данные: Функция должна возвращать результат заданного типа данных из ключевого узла</p>		

3.1.4 Удаление ключа и связанных с ним данных

Функция: DELETE KEY AND DATA	ID: DDC23-04	Интерфейс:
<p>Входные данные: Функция должна принимать в качестве входных данных один аргумент:</p> <ul style="list-style-type: none">Литерал удаляемого ключа <p>Обработка данных: Перед удалением ключа необходимо провести проверку на наличие ключа, с помощью функции поиска ключа DDC23-02. При положительном результате функция должна начать операцию удаления ключа и ассоциированных данных из контейнера.</p> <p>Операции с данными: Удаление ключа из контейнера должно происходить от вершины ветви к корню. Если последний литерал ключа соответствует последнему узлу ветви, то удаление начинается с вершины ветви. Если в ветви есть узлы с флагами других ключей, то удаление узлов ветви прекращается на этих узлах. Таким образом ветвь становится короче, и заканчивается узлом с флагом, другого, более короткого ключа. Если в ветви отсутствуют другие узлы, то ветвь удаляется из контейнера полностью.</p>  <p>Если последний литерал ключа не достигает вершины ветви, то в соответствующем узле снимается флаг ключа и удаляются/обнуляются ассоциированный данные. В удалении узлов нет необходимости. Пример: ветвь «ПАРОМЩИК» с тремя ключами «ПАР», «ПАРОМ», «ПАРОМЩИК». При удалении ключа «ПАРОМ» - с узла «М» снимается флаг ключа и удаляются данные. Ветвь остается той же длины, но теперь содержит только два ключа «ПАР» и «ПАРОМЩИК».</p> 		
<p>Выходные данные: Функция должна возвращать результат булевского типа</p>		

3.1.5 Редактирование данных связанных с ключом

Функция: EDIT DATA	ID: DDC23-05	Интерфейс:
<p>Входные данные: Функция должна принимать в качестве входных данных, два обязательных аргумента:</p> <ul style="list-style-type: none"> Литерал редактируемого ключа Экземпляр данных, соответствующий задекларированному пользователем типу данных <p>Обработка данных: Функция должна выполнять обход ветвей контейнера также, как и в функции поиска ключа DDC23-02.</p> <p>Операции с данными: В случае если ключ найден, функция должна заменить ассоциированные данные в узле на новые и вернуть положительный результат. В случае если ключ не найден, функция должна вернуть отрицательный результат.</p> <p>Выходные данные: Функция должна возвращать результат булевского типа</p>		

3.1.6 Вывод списка всех ключей

Функция: PRINT KEYS	ID: DDC23-06	Интерфейс:
<p>Входные данные: Не требуется.</p> <p>Обработка данных: Функция должна выполнять операцию обхода всех узлов контейнера без исключения, от корня к ветвям, по аналогу функции поиска ключа DDC23-02. Каждый найденный ключ должен быть внесен в результирующий контейнер списка.</p> <p>Выходные данные: Функция должна возвращать контейнер содержащий список всех ключей контейнера в виде литералов. Для вывода рекомендуется использовать контейнер данных <code>vector<string></code></p>		

3.1.7 Вывод карты ключей и ассоциированных данных

Функция: PRINT KEYS AND DATA	ID: DDC23-07	Интерфейс:
<p>Входные данные: Не требуется.</p> <p>Обработка данных: Функция должна выполнять операцию обхода всех узлов контейнера без исключения, от корня к ветвям, по аналогу функции поиска ключа DDC23-02. Каждый найденный ключ должен быть внесен в результирующий ассоциативный контейнер в поле ключа. Для каждого найденного ключа, должны быть запрошены значения ассоциированных данных и помещены в результирующий ассоциативный контейнер в поле данных, в соответствии с ключом.</p> <p>Выходные данные: Функция должна возвращать ассоциативный контейнер содержащий список всех ключей контейнера в виде литералов и всех ассоциированных данных заданного типа. Для вывода рекомендуется использовать контейнер данных <code>map<string, T></code></p>		

3.1.8 Редактирование ключа

Функция: EDIT KEY	ID: DDC23-08	Интерфейс:
<p>Входные данные: Функция должна принимать в качестве входных данных, два обязательных аргумента:</p> <ul style="list-style-type: none">• Текущий литерал редактируемого ключа• Новый литерал редактируемого ключа <p>Обработка данных: Функция должна проверить не являются ли входные данными идентичными. Если входные данные идентичные, выполнение функции прерывается. Функция должна выполнить проверку на наличие редактируемого ключа с помощью DDC23-02. Если ключ не существует, функция должна возвращать отрицательный результат. Если ключ существует, необходимо получить ассоциированные данные ключа с помощью DDC23-03. Далее текущий ключ удаляется с помощью DDC23-04 и с помощью DDC23-01 добавляется новое значение ключа.</p> <p>Выходные данные: Функция должна возвращать результат булевского типа</p>		

3.1.9 Очистка контейнера

Функция: CLEAR	ID: DDC23-09	Интерфейс:
<p>Входные данные: Не требуется.</p> <p>Обработка данных: Функция обходит все узлы дерева от вершин к корню, и удаляет каждый узел в контейнере. Корень контейнера полностью очищается от данных.</p> <p>Выходные данные: Функция должна возвращать результат булевского типа</p>		

3.2 Эксплуатационные (нефункциональные) требования

критерии оценки важных параметров работы системы

3.2.1 Производительность

Получение данных по заданному литералу, удаление и опрос на соответствие ключа, должно иметь линейную сложность $O(N)$.

3.2.2 Сохранность данных

Контейнер не должен допускать утечку памяти.

Контейнер не должен допускать утерю ассоциативных данных.

3.2.3 Безопасность

Множественные модификационные операции с одним и тем же ключом, не должны приводить к нарушению целостности контейнера, сбоя работы операторов и ошибок в использовании памяти.

4 Модели

В процессе описания спецификации, подробно представлена и описана древовидная модель контейнера. Дальнейших моделей для реализации не требуется.