Курсовая работа (проект) по фундаментальным алгоритмам (2022-2023 уч. г.)

0. (50 баллов) На языке программирования C++ (стандарт C++14 и выше) реализуйте приложение, позволяющее выполнять операции над коллекциями данных заданных типов (типы обслуживаемых объектов данных определяются вариантом) и контекстами их хранения (коллекциями данных).

Коллекция данных описывается набором строковых параметров (набор параметров однозначно идентифицирует коллекцию данных):

- название пула схем данных, хранящего схемы данных;
- название схемы данных, хранящей коллекции данных;
- название коллекции данных.

Коллекция данных представляет собой ассоциативный контейнер (конкретная реализация определяется вариантом), в котором каждый объект данных соответствует некоторому уникальному ключу. Для ассоциативного контейнера необходимо вынести интерфейсную часть (в виде абстрактного класса C++) и реализовать этот интерфейс. Взаимодействие с коллекцией объектов происходит посредством выполнения одной из операций над ней:

- добавление новой записи по ключу;
- чтение записи по её ключу;
- чтение набора записей с ключами из диапазона $[minbound \ldots maxbound]$;
- обновление данных для записи по ключу;
- удаление существующей записи по ключу.

Во время работы приложения возможно выполнение также следующих операций:

- добавление/удаление пулов данных;
- добавление/удаление схем данных для заданного пула данных;
- добавление/удаление коллекций данных для заданной схемы данных заданного пула данных.

Поток команд, выполняемых в рамках работы приложения, поступает из файла, путь к которому подаётся в качестве аргумента командной строки. Формат команд в файле определите самостоятельно.

Дополнительные задания:

За выполнение дополнительных заданий можно получить баллы. Перед описанием задания в квадратных скобках указываются номера заданий, которые необходимо выполнить для выполнения текущего. Частичная (неполная) реализация дополнительных заданий **не допускается** (за исключением заданий 6, 7, 9).

1. [0] (5 баллов) Реализуйте интерактивный диалог с пользователем. Пользователь при этом может вводить конкретные команды (формат ввода определите самостоятельно) и подавать на вход файлы с потоком команд.

2. [0] (15 баллов) Реализуйте механизм, позволяющий выполнять запросы к данным в рамках коллекции данных на заданный момент времени (дата и время, для которых нужно вернуть актуальную версию данных, передаётся как параметр). Для реализации используйте поведенческие паттерны проектирования “Команда” и “Цепочка обязанностей”.

3. [0] (10 баллов) Реализуйте механизм, позволяющий выполнять эффективный поиск по различным отношениям порядка на пространстве данных (дублирование объектов данных при этом запрещается). Обеспечьте поиск при помощи указания ключа отношения порядка (в виде строки, подаваемой как параметр поиска).
4. [0] (10 баллов) Обеспечьте хранение объектов строк, размещённых в объектах данных, на основе структурного паттерна проектирования “Приспособленец”. Дублирования объектов строк для разных объектов (независимо от контекста хранения) при этом запрещены. Доступ к строковому пулу обеспечьте на основе порождающего паттерна проектирования “Одиночка”.
5. [0] (10 баллов) Реализуйте механизмы сохранения состояния системы хранения данных в файловую систему и восстановления состояния системы хранения данных из файловой системы.
6. [0] (20 баллов) Реализуйте возможность кастомизации (при создании) реализаций ассоциативных контейнеров, репрезентирующих коллекции данных (по 5 баллов за каждую реализацию, кроме определённой Вашим вариантом): АВЛ-дерево, красно-чёрное дерево, косое дерево, B-дерево, B^+-дерево.
7. [0] (15 баллов) Реализуйте возможность кастомизации (для заданного пула схем) аллокаторов для размещения объектов данных (по 5 баллов за каждую реализацию, кроме выделения из глобальной кучи): первый + лучший + худший подходящий + освобождение в рассортированном списке, первый + лучший + худший подходящий + освобождение с дескрипторами границ, система двойников.
8. [0] (15 баллов) Реализуйте функционал приложения в виде сервера, запросы на который поступают из клиентских приложений. При этом взаимодействие клиентских приложений с серверным должно быть реализовано посредством средств межпроцессного взаимодействия (IPC). Используемые средства IPC и операционная система определяются вариантом.
9. [0, 8] (25 баллов) Реализуйте возможность кастомизации используемых средств IPC и их использование для ОС семейств Windows и Unix (по 5 баллов за каждую реализацию, кроме определённой Вашим вариантом): Unix message queues, Unix shared memory + Unix semaphores, Unix file mapping, Windows message queues, Windows shared memory + Windows semaphores, Windows file mapping.
10. [0, 8] (15 баллов) Реализуйте механизмы регистрации авторизации пользователя в системе (на клиентской стороне) и открытия пользовательской сессии (на серверной стороне) через пару значений <логин, пароль>. Пароль при этом должен храниться и передаваться в виде хеша (используйте хеш-функцию SHA-4). Логины пользователей уникальны в рамках системы, могут содержать только символы букв и цифр в количестве [5..15] (обеспечьте валидацию на стороне сервера). Пароль должен содержать не менее 8 символов (обеспечьте валидацию на стороне клиента). Формат хранения и передачи данных для авторизации пользователей определите самостоятельно.
11. [0, 8, 10] (10 баллов) На основе передаваемого в клиентские запросы токена аутентификации реализуйте различные роли, разграничивающие доступ к выполнению операций в рамках системы:

- администратор - имеет возможности создания новых пользователей; управления (выдача, блокировка) ролями других пользователей (кроме администраторов); управления доступом к схемам данных и доступом к операциям на уровне схем данных для заданных коллекций данных (режимы “только для чтения” и “чтение и модификация”);

- редактор - имеет возможности управления пулами (добавление/удаление), схемами (добавление/удаление) данных, коллекциями данных (добавление/удаление) в соответствии с предоставленными правами доступа;
- пользователь - имеет возможности взаимодействия с коллекциями данных в соответствии с предоставленными правами доступа.

12. [0, 8] (35 баллов) Реализуйте комплекс серверных приложений (далее: кластер), одно из которых (далее: entrypoint) обрабатывает входящие пользовательские запросы и делегирует их остальным серверам (далее: storage), а остальные сервера хранят данные. Распределение данных по storage серверам должно быть примерно равномерным по памяти в произвольный момент времени (используйте хеширование и “распределённый” метод цепочек разрешения коллизий).
13. [0, 8] (15 баллов) реализуйте механизм асинхронной обработки запросов (результатом запроса на выполнение операции должен являться идентификатор запроса (используйте формат GUID v4), по которому впоследствии должна иметься возможность получения результатов запроса).
14. [0, 8, 12] (25 баллов) Реализуйте децентрализованную систему обработки запросов (любой запрос может быть отправлен на любой из узлов кластера; результат асинхронного запроса также может быть получен из любого узла кластера).
15. [0, 8] (15 баллов) На стороне серверной части реализуйте механизм сохранения состояния серверов системы хранения данных в файловую систему и восстановления в набор исполняемых серверных приложений из файловой системы.
16. [0] (10 баллов) Реализуйте серверное приложение, собирающее логи клиентской (и, если есть, серверной) части приложения в файловые потоки вывода. Конфигурирование серверного логгера обеспечьте на основе файла со структурой JSON.

Для получения положительной (3 и выше) оценки за курсовую работу необходимо подготовить и сдать на кафедру пояснительную записку. В пояснительной записке необходимо:

- описать архитектуру своего приложения
- описать использованные средства языка C++
- описать использованные структуры данных и алгоритмы, их внутреннее устройство
- описать решений выполненных заданий
- привести полный исходный код реализованных заданий

Во время защиты курсовой работы необходимо уметь ориентироваться в коде, демонстрировать работу реализованного комплекса приложений, быть готовым отвечать на вопросы по языку программирования C++ и по алгоритмам и структурам данных.
Оценивается **только** финальная версия реализованного комплекса приложений.

Оценка 3 выставляется за 125-174 набранных баллов.
Оценка 4 выставляется за 175-239 набранных баллов.
Оценка 5 выставляется за 240-300 набранных баллов.

Варианты типов данных (**полужирным шрифтом** выделены поля, формирующие уникальный ключ объекта данных):

1. Данные о доставке (**id пользователя**, **id доставки**, описание доставки, ФИО пользователя (раздельные поля), адрес электронной почты пользователя, номер телефона пользователя, адрес доставки (в виде строки), комментарий пользователя, дата/время доставки)
2. Данные игрока MMO (**id пользователя**, никнейм, **игровая зона (строка)**, статус (обычный игрок/премиум игрок/модератор чата/администратор), значение внутриигровой валюты, значение внутриигровой премиальной валюты, количество очков опыта, дата регистрации, время проведенное в игре (в минутах))
3. Информация о прохождении контеста соискателем (**id соискателя**, ФИО соискателя (раздельные поля), дата рождения соискателя, ссылка на резюме соискателя, id закреплённого за соискателем HR-менеджера, **id контеста**, ЯП на котором реализуются задачи контеста, кол-во задач в контесте, кол-во решённых задач в контесте, было ли обнаружено списывание с микронаушником)
4. Данные о прохождении пайплайна (**id сборки**, **версия сборки**, информация о коммите с которого выполняется сборка (хеш коммита, логин разработчика, электронная почта разработчика), ссылка на сценарий сборки (путь к файлу), название сборки, информация об ошибках этапа build, информация об ошибках этапа статического анализа кода, информация об ошибках этапа прогона автотестов, ссылка на артефакты сборки (путь к директории))
5. Данные о прохождении сессии в университете (**id сессии**, **id студента**, ФИО студента (раздельные поля)), **формат отчётности (курсовая работа/зачёт/экзамен)**, **название предмета**, дата проведения зачёта/экзамена, время начала проведения зачёта/экзамена, полученная оценка (в диапазоне 2..5 для экзамена или курсовой работы, 0..1 для зачёта), ФИО преподавателя (раздельные поля))
6. Данные о встречах в рабочем календаре (**id встречи**, вид встречи (ежедневная встреча/встреча по результатам отчётного периода/собеседование/корпоратив), формат проведения (очный/дистанционный), описание встречи, ссылка на встречу в дистанционном формате, ФИО создателя встречи (раздельные поля), дата встречи, время начала встречи, продолжительность встречи (в минутах), список приглашённых на встречу (в виде строки))

Варианты курсовой работы:

1. Тип данных: 4, контейнер: B-дерево, IPC: Unix message queues
2. Тип данных: 2, контейнер: B+-дерево, IPC: Unix file mapping
3. Тип данных: 3, контейнер: косое дерево, IPC: Unix file mapping
4. Тип данных: 3, контейнер: B+-дерево, IPC: Unix message queues
5. Тип данных: 4, контейнер: АВЛ-дерево, IPC: Windows shared memory + Windows semaphores
6. Тип данных: 1, контейнер: B+-дерево, IPC: Unix shared memory + Unix semaphores
7. Тип данных: 6, контейнер: косое дерево, IPC: Unix message queues
8. Тип данных: 2, контейнер: красно-чёрное дерево, IPC: Unix file mapping
9. Тип данных: 6, контейнер: АВЛ-дерево, IPC: Windows shared memory + Windows semaphores
10. Тип данных: 5, контейнер: красно-чёрное дерево, IPC: Unix file mapping
11. Тип данных: 1, контейнер: косое дерево, IPC: Unix shared memory + Unix semaphores
12. Тип данных: 6, контейнер: АВЛ-дерево, IPC: Unix file mapping
13. Тип данных: 2, контейнер: B-дерево, IPC: Windows shared memory + Windows semaphores
14. Тип данных: 5, контейнер: косое дерево, IPC: Unix message queues
15. Тип данных: 5, контейнер: B+-дерево, IPC: Unix shared memory + Unix semaphores
16. Тип данных: 6, контейнер: АВЛ-дерево, IPC: Windows file mapping
17. Тип данных: 4, контейнер: красно-чёрное дерево, IPC: Unix message queues
18. Тип данных: 1, контейнер: B-дерево, IPC: Windows file mapping
19. Тип данных: 5, контейнер: красно-чёрное дерево, IPC: Unix message queues
20. Тип данных: 1, контейнер: АВЛ-дерево, IPC: Unix message queues
21. Тип данных: 2, контейнер: B-дерево, IPC: Windows file mapping
22. Тип данных: 1, контейнер: красно-чёрное дерево, IPC: Unix message queues
23. Тип данных: 2, контейнер: B-дерево, IPC: Windows shared memory + Windows semaphores
24. Тип данных: 2, контейнер: АВЛ-дерево, IPC: Unix shared memory + Unix semaphores
25. Тип данных: 5, контейнер: B+-дерево, IPC: Unix shared memory + Unix semaphores
26. Тип данных: 3, контейнер: B+-дерево, IPC: Windows shared memory + Windows semaphores
27. Тип данных: 3, контейнер: косое дерево, IPC: Windows shared memory + Windows semaphores
28. Тип данных: 4, контейнер: косое дерево, IPC: Unix shared memory + Unix semaphores
29. Тип данных: 4, контейнер: красно-чёрное дерево, IPC: Unix message queues
30. Тип данных: 2, контейнер: косое дерево, IPC: Unix shared memory + Unix semaphores
31. Тип данных: 5, контейнер: АВЛ-дерево, IPC: Windows file mapping
32. Тип данных: 6, контейнер: B+-дерево, IPC: Unix message queues
33. Тип данных: 3, контейнер: B-дерево, IPC: Windows shared memory + Windows semaphores
34. Тип данных: 6, контейнер: красно-чёрное дерево, IPC: Unix file mapping
35. Тип данных: 3, контейнер: B+-дерево, IPC: Windows shared memory + Windows semaphores
36. Тип данных: 6, контейнер: B-дерево, IPC: Windows shared memory + Windows semaphores
37. Тип данных: 1, контейнер: B-дерево, IPC: Windows shared memory + Windows semaphores
38. Тип данных: 2, контейнер: красно-чёрное дерево, IPC: Windows file mapping

39. Тип данных: 3, контейнер: B+-дерево, IPC: Unix message queues
40. Тип данных: 3, контейнер: B-дерево, IPC: Windows file mapping
41. Тип данных: 4, контейнер: АВЛ-дерево, IPC: Unix message queues
42. Тип данных: 1, контейнер: красно-чёрное дерево, IPC: Unix file mapping
43. Тип данных: 5, контейнер: АВЛ-дерево, IPC: Unix message queues
44. Тип данных: 6, контейнер: косое дерево, IPC: Windows shared memory + Windows semaphores
45. Тип данных: 4, контейнер: красно-чёрное дерево, IPC: Windows shared memory + Windows semaphores
46. Тип данных: 1, контейнер: B+-дерево, IPC: Unix file mapping
47. Тип данных: 5, контейнер: АВЛ-дерево, IPC: Unix file mapping
48. Тип данных: 4, контейнер: косое дерево, IPC: Windows file mapping