Задания к работе №5 по операционным системам и архитектуре компьютера.

Все задания реализуются на языке программирования C++ (стандарт C++14 и выше). Реализованные в заданиях приложения не должны завершаться аварийно; все возникающие исключительные ситуации должны быть перехвачены и обработаны.

Во всех заданиях запрещено пользоваться функциями, позволяющими завершить выполнение приложения из произвольной точки выполнения.

Во всех заданиях при реализации необходимо разделять контексты работы с данными (поиск, сортировка, добавление/удаление, модификация и т. п.) и отправка данных в поток вывода / выгрузка данных из потока ввода.

Во всех заданиях все вводимые (с консоли, файла, командной строки) пользователем данные должны (если не сказано обратное) быть подвергнуты валидации в соответствии с типом валидируемых данных.

Во всех заданиях необходимо контролировать ситуации с невозможностью [пере]выделения памяти; во всех заданиях необходимо корректно освобождать всю выделенную динамическую память.

Все ошибки, связанные с операциями открытия системных ресурсов уровня ОС (файлы, средства синхронизации, etc.), должны быть обработаны; все открытые системные ресурсы должны быть возвращены ОС.

Реализованные компоненты должны зависеть от абстракций, а не от конкретных реализаций абстракций. Для реализованных компонентов должны быть переопределены (либо перекрыты - при обосновании) следующие механизмы классов C++: конструктор копирования, деструктор, оператор присваивания, конструктор перемещения, присваивание перемещением.

Для задач, каталоги которых в репозитории содержат папку *tests*, требуется демонстрация прохождения всех описанных тестов для реализованных компонентов. Модификация кода тестов запрещена.

1. На основе контракта *search_tree* (repo path: */associative_container/search_tree*) реализуйте родовой класс B-дерева (repo path: */associative_container/search_tree/indexing_tree/b_tree*). Конфигурирование параметра t дерева обеспечьте посредством передачи значения в конструктор объекта. Распределение вложенных в объект дерева данных организуйте через объект аллокатора, внедряемый в объект дерева по указателю интерфейсного типа через конструктор. В узлах дерева запрещено хранение указателя на родительский узел. При невозможности выполнения операции, генерируйте исключительную ситуацию (типы исключительных ситуаций являются nested по отношению к типу дерева поиска общего вида: insertion_of_existent_key_attempt_exception, obtaining_of_nonexistent_key_attempt_exception, disposal_of_nonexistent_key_attempt_exception).

2. На основе контракта *search_tree* (repo path: */associative_container/search_tree*) реализуйте родовой класс B^*-дерева (repo path: */associative_container/search_tree/indexing_tree/b_star_tree*). Конфигурирование параметра t дерева обеспечьте посредством передачи значения в конструктор объекта. Распределение вложенных в объект дерева данных организуйте через объект аллокатора, внедряемый в объект дерева по указателю интерфейсного типа через конструктор. В узлах дерева запрещено хранение указателя на родительский узел. При невозможности выполнения операции, генерируйте исключительную ситуацию (типы исключительных ситуаций являются nested по отношению к типу дерева поиска общего вида: insertion_of_existent_key_attempt_exception, obtaining_of_nonexistent_key_attempt_exception, disposal_of_nonexistent_key_attempt_exception).

3. На основе контракта *search_tree* (repo path: */associative_container/search_tree*) реализуйте родовой класс B^+-дерева (repo path: */associative_container/search_tree/indexing_tree/b_plus_tree*). Конфигурирование параметра t дерева обеспечьте посредством передачи значения в конструктор объекта. Распределение вложенных в объект дерева данных организуйте через объект аллокатора, внедряемый в объект дерева по указателю интерфейсного типа через конструктор. В узлах дерева запрещено хранение указателя на родительский узел. При невозможности выполнения операции, генерируйте исключительную ситуацию (типы исключительных ситуаций являются nested по отношению к типу дерева поиска общего вида: insertion_of_existent_key_attempt_exception, obtaining_of_nonexistent_key_attempt_exception, disposal_of_nonexistent_key_attempt_exception).

4. На основе контракта *search_tree* (repo path: */associative_container/search_tree*) реализуйте родовой класс B^{*+}-дерева (repo path: */associative_container/search_tree/indexing_tree/b_star_plus_tree*). Конфигурирование параметра t дерева обеспечьте посредством передачи значения в конструктор объекта. Распределение вложенных в объект дерева данных организуйте через объект аллокатора, внедряемый в объект дерева по указателю интерфейсного типа через конструктор. В узлах дерева запрещено хранение указателя на родительский узел. При невозможности выполнения операции, генерируйте исключительную ситуацию (типы исключительных ситуаций являются nested по отношению к типу дерева поиска общего вида: insertion_of_existent_key_attempt_exception, obtaining_of_nonexistent_key_attempt_exception, disposal_of_nonexistent_key_attempt_exception).