

Задания к работе №3 по Системному Программированию.

Все задания реализуются на языке программирования C++ (стандарт C++20 и выше).

Реализованные в заданиях приложения не должны завершаться аварийно; все возникающие исключительные ситуации должны быть перехвачены и обработаны.

Во всех заданиях запрещено использование: глобальных переменных (включая `errno`), оператора безусловного перехода (`goto`).

Во всех заданиях запрещено пользоваться функциями, позволяющими завершить выполнение приложения из произвольной точки выполнения, вне контекста исполнения функции `main`.

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 параметры функций и вводимые (с консоли, файла, командной строки) пользователем данные должны подвергаться валидации в соответствии с типом валидируемых данных, если не сказано обратное; валидация должна зависеть от типа данных и логики применения этих данных для выполнения целевой подзадачи. При передаче аргументов приложению в командную строку, их количество также должно валидироваться.

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.

Во всех заданиях запрещено использование глобальных переменных. Во всех заданиях при реализации функций необходимо обеспечить возможность обработки ошибок различных типов на уровне вызывающего кода.

Во всех заданиях сравнение (на предмет эквивалентности или отношения порядка) вещественных чисел на уровне функции должно использовать значение эпсилон, которое является параметром этой функции.

Во всех заданиях при реализации функций необходимо максимально ограничивать возможность модификации (если она не подразумевается) передаваемых в функцию параметров (используйте ключевое слово `const`), а также объекта, в случае метода.

Для реализованных компонентов должны быть переопределены (либо перекрыты - при обосновании) следующие механизмы классов C++: конструктор копирования, деструктор, оператор присваивания, конструктор перемещения, присваивание перемещением.

Во всех заданиях необходимо уменьшать количество копирований нетривиально копируемых объектов.

Во всех заданиях необходимо проектировать компоненты с учетом SOLID принципов. Компонент не должен управлять ресурсом, если это не является его единственной задачей.

Запрещается пользоваться элементами стандартной библиотеки Си, если существует их аналог в стандартной библиотеке языка C++.

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

1. Реализуйте класс *B*-дерева (repo path: `/associative_container/search_tree/indexing_tree/b_tree`). Конфигурирование параметра t дерева обеспечьте посредством передачи значения в шаблонный параметр. Распределение вложенных в объект дерева данных организуйте через объект аллокатора, внедряемый в объект дерева 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 реализуемого компонента).

2. Реализуйте класс B^* -дерева (repo path: `/associative_container/search_tree/indexing_tree/b_star_tree`). Конфигурирование параметра t дерева обеспечьте посредством передачи значения в шаблонный параметр. Распределение вложенных в объект дерева данных организуйте через объект аллокатора, внедряемый в объект дерева 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 реализуемого компонента).

3. Реализуйте класс B^+ -дерева (repo path: `/associative_container/search_tree/indexing_tree/b_plus_tree`). Конфигурирование параметра t дерева обеспечьте посредством передачи значения в шаблонный параметр. Распределение вложенных в объект дерева данных организуйте через объект аллокатора, внедряемый в объект дерева 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 реализуемого компонента).

4. Реализуйте класс B^{*+} -дерева (repo path: `/associative_container/search_tree/indexing_tree/b_star_plus_tree`). Конфигурирование параметра t дерева обеспечьте посредством передачи значения в шаблонный параметр. Распределение вложенных в объект дерева данных организуйте через объект аллокатора, внедряемый в объект дерева 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 реализуемого компонента).

5. Реализуйте класс *B*-дерева, сохраняющее данные в файловом хранилище (repo path: */associative_container/search_tree/indexing_tree/b_tree_disk*). Конфигурирование параметра *t* дерева обеспечьте посредством передачи значения в шаблонный параметр. Распределение вложенных в объект дерева данных организуйте через объект аллокатора, внедряемый в объект дерева 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 реализуемого компонента).