# Задание на разработку структуры данных

# Таблица  – Распределение структур данных в соответствии с вариантом

|  |  |
| --- | --- |
| **№ задания** | **Структура данных** |
| 1 | ЛОС |
| 2 | АВЛ-дерево (авторы Георгий Адельсон-Вельский и Евгений Ландис) |
| 3 | B-дерево |
| 4 | ЦОС |
| 5 | Словарь |
| 6 | Красно-чёрное дерево |
| 7 | Стек (Stack) |
| 8 | Дек |
| 9 | Очередь с приоритетами |
| 10 | Линейный двусвязный список (ЛДС) |
| 11 | Граф |
| 12 | Вектор (vector) |
| 13 | Циклический двусвязный список (ЦДС) |
| 14 | Очередь |

1. Зарегистрироваться в GitHub. Создать новый репозиторий с "говорящим именем", например, MIIT.DBMS.%LastName%.%StructureName%. Пригласить меня, MarinaPaley, в качестве колаборатора. Все дальнейшее взаимодействие проводить через Pull request и Issue.
2. Для заданной структуры данных разработать API (программный интерфейс приложения, интерфейс прикладного программирования) (англ. application programming interface) — описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой. На языке C++ написать конструктор и деструктор класса разрабатываемой структуры для типа int. Отладить программу.
3. Разработать метод вывода структуры в строку. Отладить приложение. Все изменения выложить в GitHub. Завести новый Pull request. Провести рефакторинг кода в соответствии с Issue.
4. Разработать все необходимые для заданной структуры методы для хранения и поиска (если задано) элемента типа int. Отладить приложение. Все изменения выложить в GitHub. Завести новый Pull request. Провести рефакторинг кода в соответствии с Issue.
5. Структура данных для любого типа данных, реализующая CRUD — акроним, обозначающий четыре базовые функции, используемые при работе с базами данных: создание (англ. create), чтение (read), модификация (update), удаление (delete). Изменить свою структуру данных для возможности хранения любых типов данных (использование шаблонов). Отладить приложение. Все изменения выложить в GitHub. Завести новый Pull request. Провести рефакторинг кода в соответствии с Issue.
6. Собственный итератор. Разработать собственный итератор для "прохождения" по элементам разрабатываемой структуры данных. Отладить приложение. Все изменения выложить в GitHub. Завести новый Pull request. Провести рефакторинг кода в соответствии с Issue.
7. Переопределение оператора сдвига. Переопределить операторы сдвига для реализации удобного взаимодействия с потоком ввода/вывода. Реализовать (не в библиотеке классов) методы ввода/ вывода структуры из/в консоль и файл, используя операторы сдвига. Отладить приложение. Все изменения выложить в GitHub. Завести новый Pull request. Провести рефакторинг кода в соответствии с Issue.
8. Тесты. Разработать тесты на конструктор и деструктор. Тесты положить в отдельную папку с названием Tests. Отладить приложение. Все изменения выложить в GitHub. Завести новый Pull request. Провести рефакторинг кода в соответствии с Issue.
9. Полное тестирование приложения. Разработать тесты на все публичные методы разрабатываемой структуры данных. Отладить приложение. Все изменения выложить в GitHub. Завести новый Pull request. Провести рефакторинг кода в соответствии с Issue.
10. Отчет по лабораторной работе "Разработка структуры данных" Разработать отчет по лабораторной работе по ГОСТ НИР 2017 (http://docs.cntd.ru/document/1200157208). Отчет должен содержать:
11. Задание на работу
12. UML-диаграмму классов приложения
13. Листинг готового приложения в текстовом формате. Для текста кода использовать шрифт "Courier New" или "Consolas" размером 11 пт с однострочным интервалом. Программу перед этим отформатировать.
14. Результаты работы приложения в виде снимков экрана. Все рисунки должны быть крупными и четкими, иметь подписи в соответствии с ГОСТ НИР 2017. Цвет фона рисунка должен быть белым, цвет шрифта ‑ черным.