

Практика по деревьям

Юрий Литвинов
y.litvinov@spbu.ru

21.02.2026

Задача в целом

Реализовать абстрактный тип данных «Множество» на основе бинарного дерева поиска

- ▶ Две части: на паре и дома
 - ▶ На паре можно получить 3 балла, дома 7
- ▶ Командная работа «по-взрослому»
 - ▶ Общий репозиторий (лучше создать с нуля)
 - ▶ Парное программирование на старте
 - ▶ Взаимодействие через пуллреквесты и ревью
- ▶ Команды по 2-3 человека

На паре

- ▶ Создать репозиторий для проекта
- ▶ Вместе спроектировать структуру проекта и написать общий заголовочный файл
- ▶ Описать общую структуру для дерева и его узла
 - ▶ С соблюдением правил сокрытия деталей реализации
- ▶ Решить задачи А, В и С (ниже), либо совместно за одним ноутбуком, либо параллельно
 - ▶ Узнайте, что такое co-authored-by на GitHub
- ▶ Настроить CI (с clang-format и clang-tidy)

Задача А — Структура и вставка

- ▶ Опишите абстрактный тип данных — двоичное дерево поиска (BST) — и его структуру
- ▶ Реализуйте функции:
 - ▶ **void** bstInsert(BST* tree, **int** value) — вставить значение в дерево
 - ▶ **bool** bstContains(BST* tree, **int** value) — вернуть **true**, если значение есть в дереве
 - ▶ **void** bstFree(BST* tree) — освободить всю память, занятую деревом
- ▶ Напишите main с несколькими простыми проверками: вставьте несколько элементов и убедитесь, что bstContains возвращает правильные результаты

Задача В — Обходы

- ▶ Реализуйте функции обхода дерева (рекурсивно), печатающие элементы:
 - ▶ **void** bstInorder(BST* tree)
 - ▶ **void** bstPreorder(BST* tree)
 - ▶ **void** bstPostorder(BST* tree)
- ▶ Все функции должны корректно работать на пустом дереве

Задача С — Статистика дерева

- ▶ Реализуйте функции:
 - ▶ **int** bstHeight(BST* tree) — высота дерева
 - ▶ **int** bstSize(BST* tree) — количество узлов
 - ▶ **int** bstMin(BST* tree) — минимальное значение
 - ▶ **int** bstMax(BST* tree) — максимальное значение
- ▶ Для bstMin и bstMax определите поведение на пустом дереве и задокументируйте его в заголовочном файле