数据结构与算法分析 & C++高级特性的研究

目标

1. 充分理解各数据结构的底层运行逻辑。
2. 深度掌握 C++ 的高级特性（面向对象、内存管理、模板编程等）。
3. 最终自主实现一个高质量的数据结构库（具备 STL 风格接口与工程级组织结构）。

第一阶段：C++核心基础夯实（1–2周）

1. 面向对象编程

- 类与对象：封装、继承、多态
- 构造函数与析构函数（深拷贝 vs 浅拷贝）
- 运算符重载（拷贝赋值、比较、流输出等）
- 友元函数与友元类
- 静态成员与常量成员
- **RAII（资源获取即初始化）思想**
- **C++对象模型**（this指针、虚函数表、内存布局）

2. 内存管理与拷贝控制

- 栈与堆的内存分配
- `new/delete` 与 `malloc/free` 的区别与使用规范
- 三/五法则（拷贝构造、拷贝赋值、析构、移动构造、移动赋值）
- 移动语义与右值引用（C++11+）
- 内存泄漏检测与防范（Valgrind / Sanitizer 工具）

第二阶段：基础数据结构实现（3–4周）

3. 线性结构

动态数组（Vector）

- 动态扩容策略与内存管理
- 模板类实现
- 异常安全保证（强/基本/无异常保证）
- 迭代器实现（随机访问迭代器）
- 拷贝与移动构造函数优化

链表（Linked List）

- 单向、双向与循环链表实现
- 节点与迭代器封装
- 智能指针（ `unique_ptr` / `shared_ptr` ）在链表中的应用
- 异常安全性与内存释放设计

4. 栈与队列

- 基于数组与链表的栈/队列实现
- 模板编程应用
- 适配器模式（将底层容器封装为不同抽象接口）

第三阶段：中级数据结构与C++进阶（4–5周）

5. 树结构

二叉树

- 递归与非递归遍历（前中后序、层次）
- 模板类与节点泛型化设计

二叉搜索树（BST）

- 插入、删除、查找逻辑
- 异常与边界处理机制

平衡树家族

- AVL 树实现与旋转操作
- 红黑树（重点）：插入、删除、重平衡机制
- AVL vs Red-Black 性能与设计对比

6. C++高级特性深入

- 模板编程：函数模板与类模板
- 模板特化与偏特化
- **SFINAE 与 C++20 Concepts 基础**
- 智能指针： `unique_ptr` ， `shared_ptr` ， `weak_ptr` 与自定义删除器
- `decltype` ， `auto` ， `constexpr` 在泛型代码中的使用

第四阶段：高级数据结构实现（4–5周）

7. 哈希结构

- 哈希表原理与实现
- 冲突解决策略：链地址法、开放定址法
- 模板化键值存储（支持任意类型）
- 自定义哈希函数与 `std::hash` 特化机制
- `unordered_map` / `unordered_set` 风格接口

8. 堆与优先队列

- 二叉堆实现（最小堆与最大堆）
- 模板化比较函数（ `Compare` 模板参数）
- 堆排序算法
- 模板优先队列封装

9. 图结构

- 邻接矩阵与邻接表封装
- 图遍历算法（BFS、DFS）
- 最短路径算法（Dijkstra）
- 最小生成树算法（Prim / Kruskal）

- 模板化节点与边结构

第五阶段：专业级库开发（3–4周）

10. 标准库特性实现

- STL风格接口设计： `begin()` ， `end()` ， `size()` ， `empty()`
- 迭代器设计模式与分类
- **分配器（Allocator）机制**
- 异常安全保证（强、基本、无异常保证）
- 移动语义优化与性能分析

11. 性能优化与工程化

- 时间复杂度与空间复杂度分析
- 内存池设计与复用机制
- 缓存友好设计（数据局部性）
- 内联函数与编译优化（ `inline` ， `constexpr` ， `noexcept` ）

第六阶段：项目整合与进阶（2–3周）

12. 完整库架构

- 命名空间与模块划分（如 `ds::vector` ， `ds::map` ）
- 头文件组织与依赖最小化
- **CMake 构建系统基础**
  - CMakeLists.txt 编写
  - 静态库与动态库构建
  - Include / Lib 路径配置
- 测试框架集成（Google Test 或 Catch2）
- 文档生成（Doxygen）

13. 扩展主题

- 并发安全数据结构（互斥锁、原子操作）
- 自定义分配器与对象池
- STL兼容接口与算法适配
- C++20 模块化与 Concepts 扩展

实践项目里程碑

1. 实现基础容器（ `Vector` 、 `LinkedList` ）
2. 完成树结构家族（ `BST` 、 `AVL` 、 `RedBlackTree` ）
3. 构建哈希容器系列（ `HashMap` 、 `HashSet` ）
4. 实现堆与图算法库
5. 整合成统一的数据结构库 `libds`
6. 使用 CMake + 测试框架构建工程
7. 生成完整文档（Doxygen）