复习要点

Ch1

1. 数据结构相关的基本概念
2. 面向对象的基本概念和原则，**The Open-Closed Principle, Subclass Substitution Rule**
3. 编程基础（**C++**）

Ch2

1. 容器的基本概念
2. 基于连续结构（**array**）和链接结构（**linked-list**）实现基本的容器的原理
3. 迭代器的概念与迭代器的实现

Ch3

1. 算法时间复杂度的概念
2. 代码段和常见算法时间复杂度 **WorstTime(n)**的计算

Ch4

1. 递归的基本概念、递归函数的基本形式和执行过程
2. 简单递归算法的编程实现
3. 递归的时间复杂度

Ch5

1. **vector** 和 **deque** 的定义和实现原理（逻辑结构和存储结构）
2. **vector, deque** 容器类的使用（声明对象，调用方法，存储元素）
3. 使用 **vector, deque** 实现特定的算法

Ch6

1. **list** 的定义和实现原理（逻辑结构和存储结构）
2. **list** 容器类的使用（声明对象，调用方法，存储元素）
3. 使用 **list** 实现特定的算法

Ch7

1. 容器适配器（**container adapter**）的概念和实现原理
2. **queue** 和 **stack** 的定义和实现原理（逻辑结构和存储结构）
3. **queue, stack** 容器类的使用（声明对象，调用方法，存储元素）
4. 应用：使用 **queue, stack** 实现特定的算法：利用队列实现排队，利用栈实现表达式的转换（**infix->>postfix**，**prefix**）、递归程序改写为非递归程序
5. 应用 **container adapter** 思想改造已有容器

Ch8

1. 二叉树（**binary tree**）的基本概念及相关概念（**root, leaf, path, height, depth, child, parent…**）
2. 二叉树常见操作的递归实现（计算树中的节点数、叶子、树高**…**）
3. 特殊二叉树的概念、性质及公式（**two-tree, full-tree, complete-tree**）
4. 二叉树的遍历及操作的实现（**pre-order, in-order, post-order, breadth-first**）
5. 二叉搜索树（**binary search tree**）的概念和实现原理（**BinSearchTree class**）
6. 二叉搜索树的查找、插入、删除基本操作算法。

Ch9

1. 平衡的二叉树的概念
2. **AVL** 树的基本概念
3. 二叉树的四种旋转操作过程

Ch10

1. 红黑树**(red-black tree)**的基本概念
2. 红黑树的查找、插入、删除操作过程
3. 基于红黑树实现的 **4** 种容器类**(map, set, multi-set, multi-map)**的作用
4. **map, set** 容器的声明和操作使用

Ch11

1. 优先级队列的概念和实现方式（基于堆）
2. 堆的概念（上下有序的二叉树）和堆的存储实现（数组）
3. 堆的插入、删除操作过程

Ch13

1. 哈希（**hash**）的概念和作用，哈希处理核心内容（哈希函数和冲突处理机制）
2. 哈希函数的面向对象实现（函数类）
3. 哈希冲突处理机制（链式哈希和开放地址哈希-双哈希）的原理。

《 类库与数据结构》考试题型

I. Answer the following questions.（10 items, 3 Points for each, 30 points total）

1. What is a container adapter? Give an illustration.

2. Using C++ to declare(声明) a iterator of the list whose item type is the string.

3. Using C++ to declare a set object *BATT*, the type of each item is integer and order is decreasing order (降序).

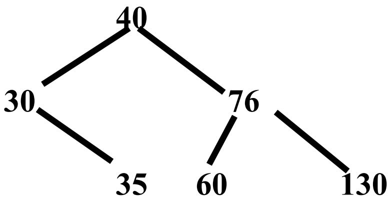
4. Write he recursive algorithm(递归算法) of height(t) which calculate(计算) the height of a binary tree t.

5. What is the WorstTime(n) (时间复杂度) of loop:

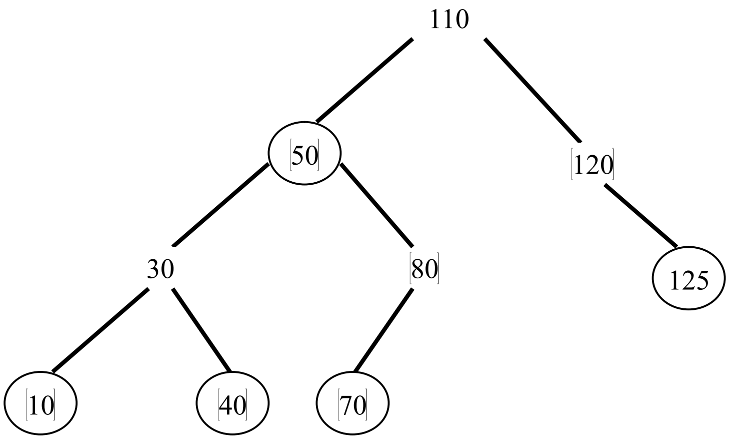
```
for(i=1; i<=n; i+=2)
   sum+=i;
```

II. Analysis and draw diagrams(4 items, 5 points for each, 20 points total)

1. Delete the 76 from the following Binary Search Tree, draw the diagram of the new tree.



2. Insert 130 into the following red-black tree (the circled nodes are red), and fix after insertion, draw the diagram of the new tree.



III. Read the programs and answer the questions (4 items, 5 points for each, 20 points total)

1. **Suppose the input is following:**

```
Red
Blue
White
Grey
Brown
***

Code segment:
BinSearchTree<string> words;
BinSearchTree<string>::Iterator itr1;
string word;
cout << "Please enter a word; the sentinel is ***: ";
cin >> word;
while (word != "***")
{
   words.insert (word);
   cout << "Please enter a word; the sentinel is ***: ";
   cin >> word;
} // while
for (itr1 = words.begin( ); itr1 != words.end( ); itr1++)
  cout << *itr1 << endl;
words.erase (words.find ("Grey"));
words.erase (words.begin( ));
for (itr1 = words.begin( ); itr1 != words.end( ); itr1++)
  cout << *itr1 << endl;
```

IV. Algorithm Description or proof (10 points)

1. **Describe implement structure (实现结构) of the list container in STL, and describe the main step of the insert operation of the list.**

2. **Try to proof: As h is the height of a full tree T, the number of items in T is 2^{h+1}-1.**

V. Programming（2 items, 20 points total）

1. **Try to implement a stack based on container provided in STL, whose push and pop operation at the front of container. (10 points)**

2. **Try to implement a recursive method preorder_traversal of the BinSearch tree class, which accesses the binary search tree in preorder(先序遍历). （10 points）**