**第三次实验**

**注意事项**

(1) 不是每个作业题都需要提交，注意审题

**实验六、数组、指针与字符串**

**选做**：(不需要提交)

(1) 编写并测试3×3矩阵转置函数，使用数组保存3×3矩阵。

(2) 使用动态内存分配生成动态数组来重新完成第(1)题，使用指针实现函数的功能。

(3) 编程实现两字符串的连接。要求使用字符数组保存字符串，不要使用系统函数。

(4) 使用string类声明字符串对象，重新实现第(3)题。

(5) 声明一个Employee类，其中包括姓名、街道地址、城市和邮编等属性，以及change\_name()和display()等函数。display()显示姓名、街道地址、城市和邮编等属性，change\_name()改变对象的姓名属性，实现并测试这个类。

(6) 声明包含5个元素的对象数组，每个元素都是Employee类型的对象，测试该对象数组并展示相关信息。

(7) 搜索字符串：第一次输入一串较长的字符，第二次输入一串较短的字符，在长串中查找短串，输出：Yes或No，要求查找时不区分大小写，使用字符数组保存字符串，不使用系统函数，不使用string类。

**必做1**：

实现一个针对Student结构的双向队列容器。相关代码已由老师提供，请填充其中空缺的部分。双向队列容器的示意图如下，请结合StudentDeque.h文件查看。

后边际数组

前边际数组

rootArr

Student元素数组

rootBackIndex

rootFrontIndex

指针数组

rootCapacity表示本数组的容量

indexOfBackArr

indexOfFrontArr

相关说明：

1）用户使用该容器时，通过pushBack()、pushFront()从后端、前端插入元素，或者通过popBack()、popFront()从后端、前端把最外面的元素弹出来，所以操作总是发生在前、后边际数组上，对应rootArr数组中rootBackIndex和rootFrontIndex两个下标。

2）注意后边际数组从末尾向前进行填充。

3）indexOfBackArr是后边际数组最外侧元素的下标，如果indexOfBackArr为负值，则表明后边际数组中暂无元素，indexOfFrontArr同理。

4）从后端插入元素的时候，如果当前的后边际数组已经塞满，就需要通过调用addBackArr()新增一个Student元素数组，作为新的后边际数组，然后再插入元素，addFrontArr()同理。

5）从后端把最外面的元素弹出来的时候，如果当前的后边际数组只剩最后一个元素，则弹出后需要调用removeBackArr()把清空的数组删除，并将rootBackIndex加1（即把右边相邻那个Student元素数组当作新的后边际数组），removeFrontArr()同理。

6）容器在两个方向都会增长，如果rootArr数组的某一边已塞满，再插入新元素时就要调用expandRoot()扩容一倍，扩容后要把原来的内容调整到rootArr数组的中间。

7）特殊情况1：如果还剩下唯一一个Student元素数组，则既是后边际数组，也是前边际数组，此时如果indexOfBackArr或indexOfFrontArr为负，表明整个容器没有一个元素，那么插入元素时，无论是pushBack()还是pushFront()，都须把元素插入中间位置，而非两端。

8）特殊情况2：如果还剩下唯一一个Student元素数组，则既是后边际数组，也是前边际数组，那么即使其元素清空了，也要保留该数组而不删除。

9）程序中多次出现“>> 1”，意思是右移一位，即除以2。

Deque的两种常见应用：如果总是从某一端push，从另一端pop，Deque就变成单向的队列，实现排队功能；如果总是从某一端push，从同一端pop，Deque就变成栈。

提交时，除了关键代码，还要提交#define LARGE\_AMOUNT注释掉之前和之后的运行结果截图。

**必做2**：

实现一个针对Student结构的散列（Hash）集合容器。相关代码已由老师提供，请填充其中空缺的部分。Hash集合容器的示意图如下，请结合StudentHashset.h文件查看。

rootArr

指针数组，rootCapacity表示本数组的容量

totalCount表示容器内所有元素的数量

相关说明：

1）Hash值：每个Student结构体都有自己的Hash值，假定rootArr数组的容量（即rootCapacity）是16，则一个Student结构体的Hash值就是其ID除以16的余数。

2）rootArr数组是个指针数组，每个元素作为头指针组织一个链表，同一个链表的所有Student结构体拥有相同的Hash值，即对应的rootArr数组中的下标。

3）用户使用该容器时，通过insert(s)把元素s插入恰当的位置，通过find(s)，给出Student结构体s即可在容器中快速找到与s相同的元素，通过remove(s)，给出Student结构体s即可找到与s相同的元素并从其链表上删除。

4）一群随机ID的Student结构体插入容器，众多链表总会有长有短，为保证快速查找，要防止链表过长，为此规定，一旦元素总数（即totalCount）超过rootArr数组容量（即rootCapacity），就对rootArr数组扩容一倍。

5）rootArr数组一旦扩容，也就意味着所有元素的Hash值发生变化，需要全部重新计算Hash值并据此重新构建链表。

Hash集合容器的用途：用于不关心顺序，但希望快速插入、快速查找、快速删除的场合。

提交时，除了关键代码，还要提交#define LARGE\_AMOUNT注释掉之前和之后的运行结果截图。