哈尔滨工业大学（深圳）

数据结构实验报告

线性结构及其应用

学　院: 计算机科学与技术
姓　名: 胡博涵
学　号: SZ170110113
专　业: 计算机科学与技术
日　期: 2018-04-26

一、问题分析

题目的需求是以班级为单位，存储每个学生的信息，包括姓名、学号等信息，转换为计算机的问题就是，如何在计算机内存储具有同一抽象数据类型的群体数据，且该群体数据的规模并不能完全确定，因此要求需要动态地调整存储空间，成绩要求降序存储，因此也需要一个高效的排序算法，对于不同的存储结构而言，如何选择合适的排序算法也是极为关键的。

二、详细设计

2.1 设计思想

问题要求的是存储学生的成绩、学号、姓名等信息，因此在程序中定义一个 Student 结构体类型。为存储群体性的同类数据，通常有以下两种存储方式：

（1）顺序存储结构：

对 C／C++已有的数组结构进行一般性地推广，即数据存储在一段起始地址为 P 的连续的内存空间内，并且其中的每一个元素都对应于唯一的一个下标编号 i（一般从 0 开始），要访问 i 所指的那个元素，就需要访问地址为 P+sizeof(Base type)*i 单元内的元素（若是在 C++中，我们可以通过重载运算符来实现 P[i]的直接访问）。换言之，第 i 号元素的地址和其下标为线性关系。

采用这种顺序存储结构的好处在于可以在常数复杂度的时间内进行读取、修改等静态操作，缺点在于在对元素进行增加、删除等动态操作时，需要耗费较大的时间复杂度。

由于本实验采用动态数组，因此我们还需考虑重新分配空间并拷贝数据带来的开销。为将这个开销降至最低，我采取的扩容策略是：当有效容量达到总容量时，总容量自动扩容为 2 倍，即能够保证占用率（size/capacity）在 0.5 与 1 之间，容量(capacity)以指数形式增加，在扩容方面的时间复杂度为 O(2n+4n+8n+……+capacity(n)) <O(2*capacity(n)) = O(n),插入了 n 个元素，进行了 log2n 次扩容，所以单次插入分摊复杂度仅为 O(1).

（2）链式存储结构：

与顺序存储结构不同，链式存储结构虽然在逻辑上有着线性次序，却在物理存储位置上互不相临。在程序运行的过程中，我们可以通过动态申请和释放空间来达到实现需求的目的。而元素的物理存储是离散的，我们就需要通过指针的机制，建立起他们逻辑上的联系。本次实验采取的链表，就是一种典型的链式存储结构，节点之间可以通过指针相互索引。这符合我们对于“学生数不定”的动态需求。

（3）部分算法设计

由于程序的特殊性，查找算法只能选取线性查找算法。插入将与排序同时进行，即在插入元素之前，先定位元素应当插入的位置，使得元素插入之后，列表依然有序。合

并四个表格采取的是二路归并算法，将表格依次归并。归并算法并非原地工作，而是输出至另一个新表之中。

2.2 存储结构及操作

（0） 基本存储数据类型

定义“Student”结构体，成员变量为字符数组“ID”、字符数组“name”、整型变量 score.

函数名	参数	功能	返回值
initStudent()	无	创建一个“Student”对象	创建后的对象指针
printRecord(Student*)	Student 指针	输出该对象的成员内容	无

（1）顺序存储结构(Array List)

成员变量：数据域指针_elem、总容量 capacity、有效容量 size

函数名	参数	功能	返回值
init_List()	无	以默认大小 构造顺序表	初始化后 顺序表
Expand(*List)	要扩容的线性表	将线性表扩容 2 倍	无
Insert(List*,Elemtype,int)	目标线性表、待插入元素、待插入元素位置	将元素插入到指定位置	无
locate(List*,Elemtype)	目标线性表、待插入元素	定位线性表中 第一个数据大于/小于 目标数据的位置	线性表中 某个位置
insert_Ranked(List*,Elemtype)	目标线性表、待插入元素	调用 locate 函数，使得元素插入后， 线性表仍然保持有序	无
printList(List*)	目标线性表	输出线性表数据	无
searchBykey(List*,(*Fun)(Student),char)	目标线性表、函数指针、目标字符串	搜索线性表中 相应键值字符串	线性表中 某个位置

（2）链式存储结构（Linked List）

1） 节点数据类型(Node)

成员变量：数据域 Data, 地址域 next

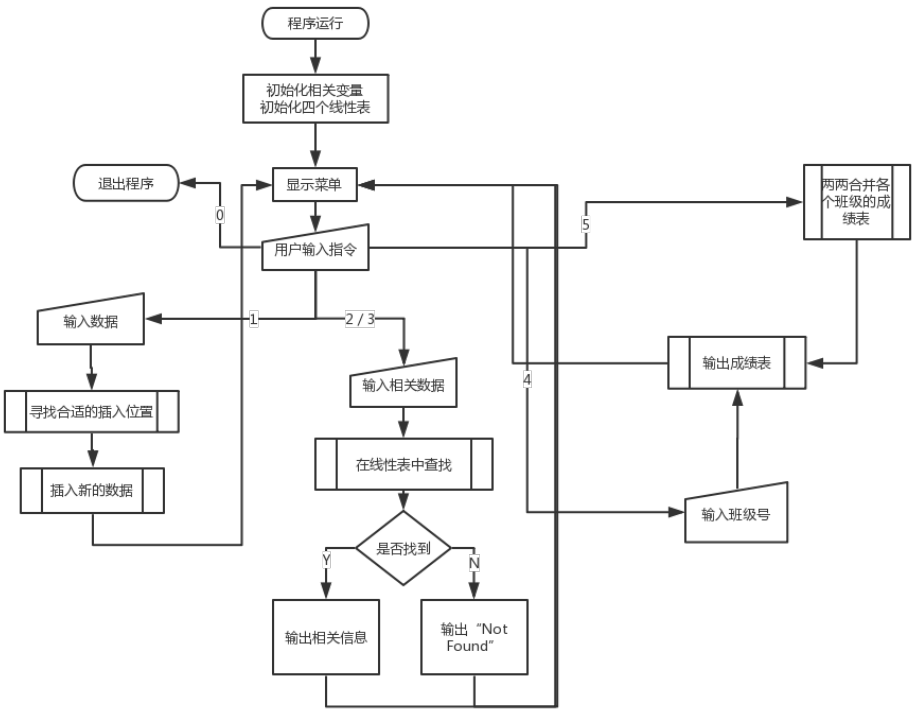
2）链表(Linked List)

成员变量：头节点 head,尾节点 tail

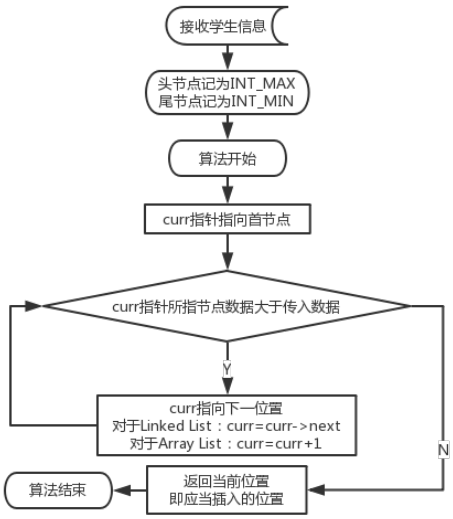
函数名	参数	功能	返回值
initList()	无	以默认大小 初始化顺序表	初始化后 顺序表
Insert(Node*,Node*)	插入节点的前驱位置、 待插入节点	将元素插入节点之后	无
locate(List*,Elemtype*)	目标线性表、待插入元 素	定位线性表中 第一个数据大于/小于 目标数据的位置	该位置的 前驱节点
insert_Ranked(List*,Elemtype)	目标线性表、待插入元 素	将待插入元素封装为节 点，调用 locate 函数，使 得元素插入后， 线性表仍然保持有序	无
printList(List*)	目标线性表	输出线性表数据	无
searchBykey(List*,(*Fun)(Student), char)	目标线性表、函数指针、 目标字符串	搜索线性表中 相应键值字符串	线性表中 某个位置

2.3 程序整体流程

总体流程：

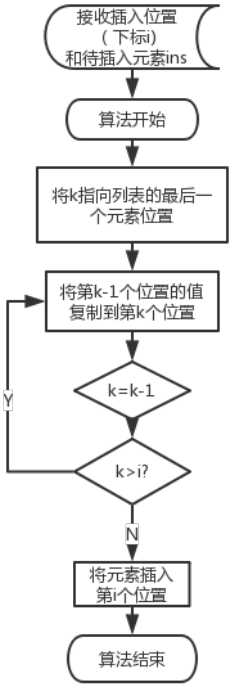


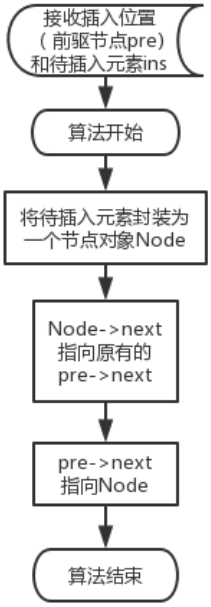
Locate 算法：查找插入位置



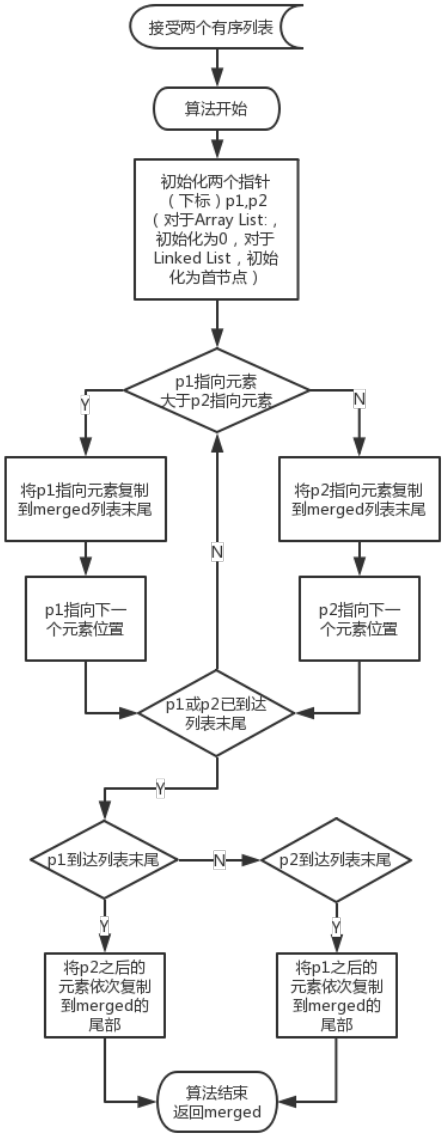
Insert 算法：

顺序表（Array List）：





二路归并算法：



三、用户手册

程序运行时，会输出一个菜单，并提示输入选项：

1.Input record

2.Search by ID

3.Search by name

4.List record of each class

5.List record of all classes

0.Exit

Please enter your choice:

录入单条成绩：输入 1 后回车，会依次询问班级、学号、姓名、成绩，其中，“班级”和“成绩”为整数类型，“姓名”和“学号”为字符数组类型。

查询某班级成绩：输入 4 后回车，会询问班级，键入班级后回车即可。

查询某学生成绩：可通过“学号”或“姓名”查询成绩，“学号”输入 2，“姓名”输入 3，会提示输入学号或姓名，输入后回车，若能查询到，则会输出学生信息，若未查询到，输出“Not Found”。

查询综合排名：输入 5 后回车，会输出四个班所有学生按成绩综合排序的结果。

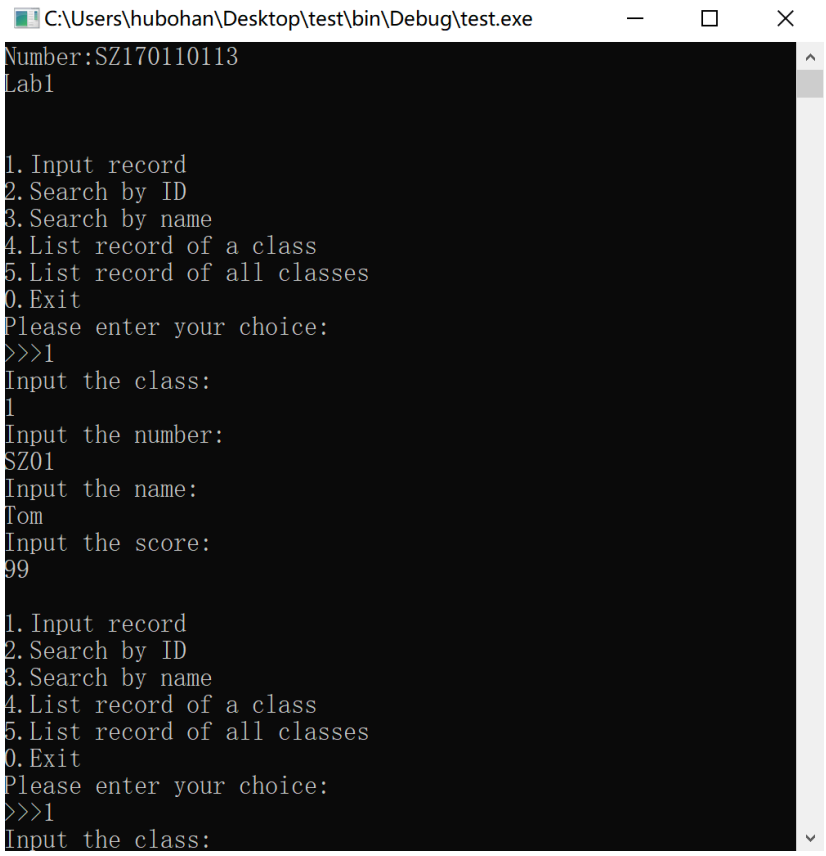
四、总结

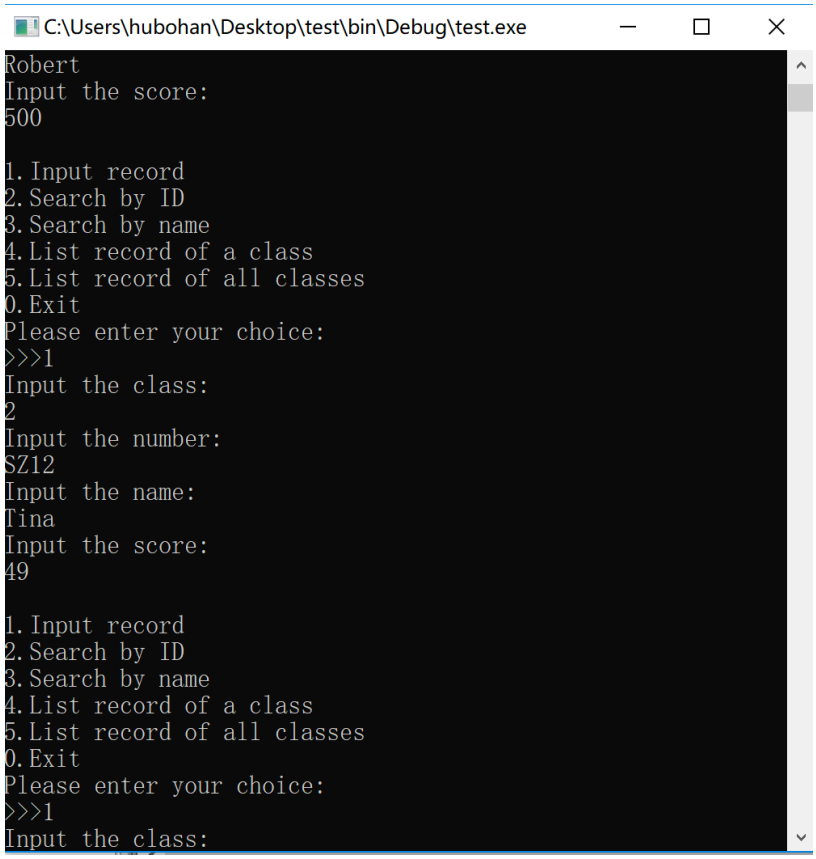
该实验涉及到的数据结构有线性表，线性表分为顺序存储的可变长数组（Array List）和链式存储结构的链表（Linked List），其中，本实验采用的是单向链表和顺序存储结构。其涉及的主要算法为插入、定位、查找、二路归并，其中，由于本次实验性质特殊性（必须按照成绩降序排列），查找算法采用了效率较低的顺序查找。定位（Locate）算法事实上就是对有序列表顺序查找的一个变式，即寻找到插入的正确位置。对于单向链表而言，插入算法效率为常数复杂度，而对于数组而言，插入算法平均复杂度为 O(n/2)，而静态操作，如访问、修改等操作，对于链表而言，不能随机访问，复杂度较高，对于数组而言，却只有常数复杂度。那么，我的问题是，是否存在某种数据结构，能够在静态操作和动态操作中取得均等的效率呢？是否树这种折中的方案就是我们的选择呢？

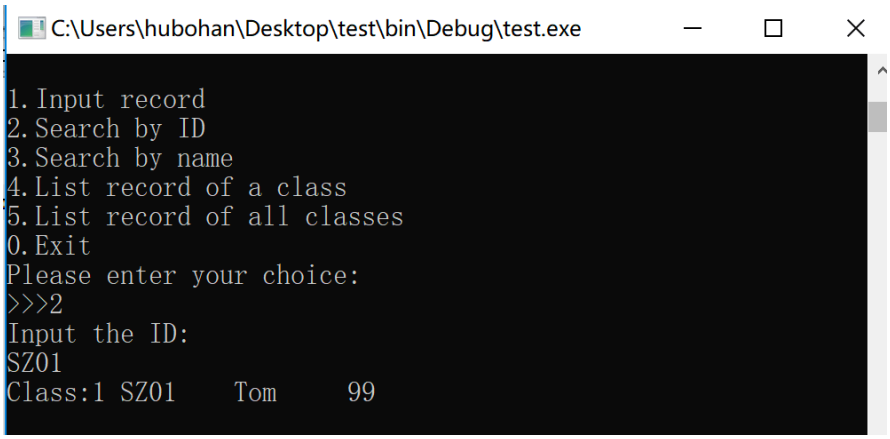
通过这次实验，我复习了 C 语言的相关知识，尤其是对指针和结构体的应用，学会了用程序语言实现 ADT，同时也锻炼了 debug 技能。

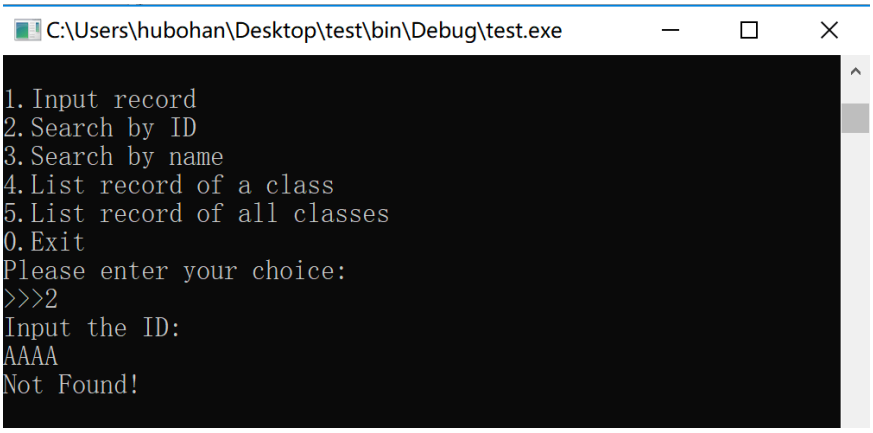
五、结果

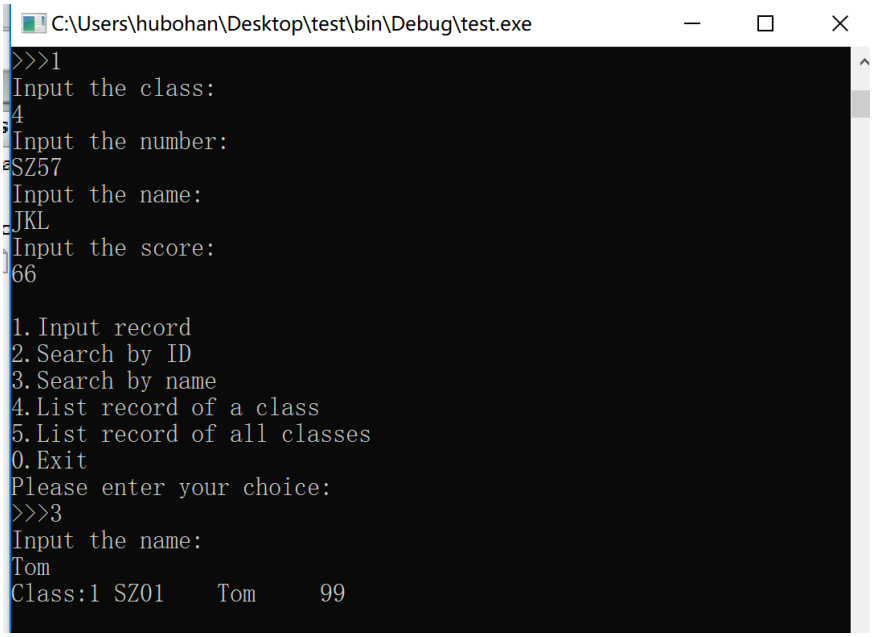
程序正确运行的结果截图。

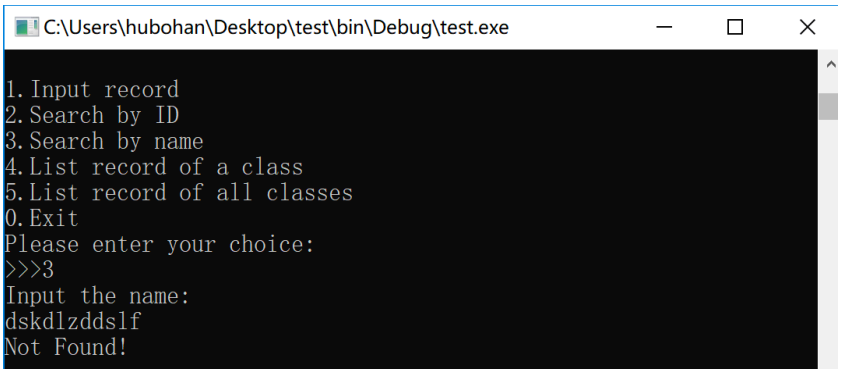

C:\Users\hubohan\Desktop\test\bin\Debug\test.exe
Number:SZ170110113
Lab1
1.Input record
2.Search by ID
3.Search by name
4.List record of a class
5.List record of all classes
0.Exit
Please enter your choice:
>>>1
Input the class:
1
Input the number:
SZ01
Input the name:
Tom
Input the score:
99
1.Input record
2.Search by ID
3.Search by name
4.List record of a class
5.List record of all classes
0.Exit
Please enter your choice:
>>>1
Input the class:

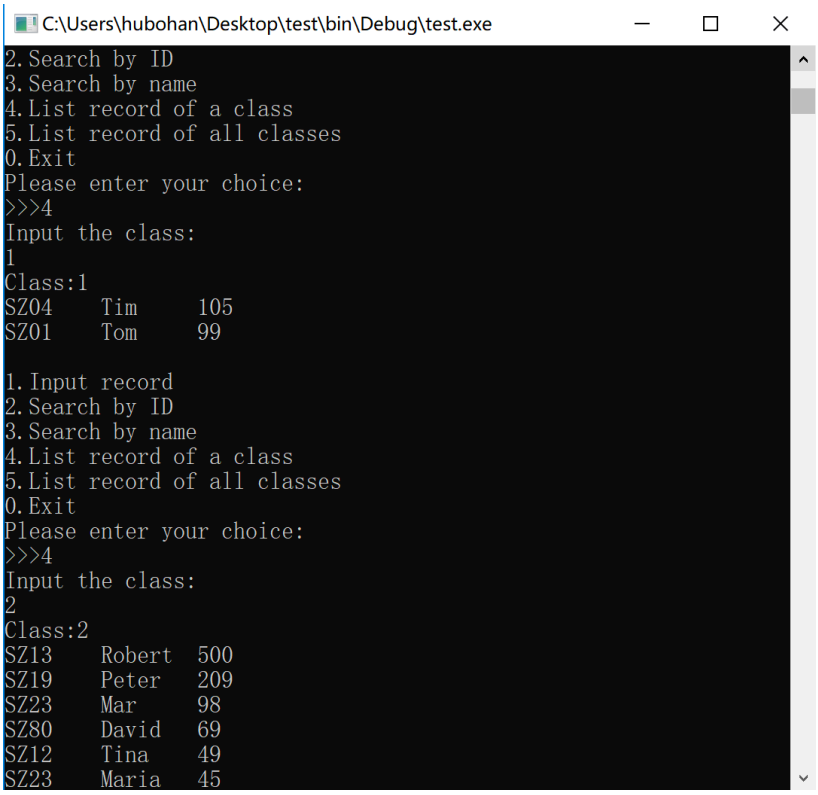

C:\Users\hubohan\Desktop\test\bin\Debug\test.exe
Robert
Input the score:
500
1.Input record
2.Search by ID
3.Search by name
4.List record of a class
5.List record of all classes
0.Exit
Please enter your choice:
>>>1
Input the class:
2
Input the number:
SZ12
Input the name:
Tina
Input the score:
49
1.Input record
2.Search by ID
3.Search by name
4.List record of a class
5.List record of all classes
0.Exit
Please enter your choice:
>>>1
Input the class:

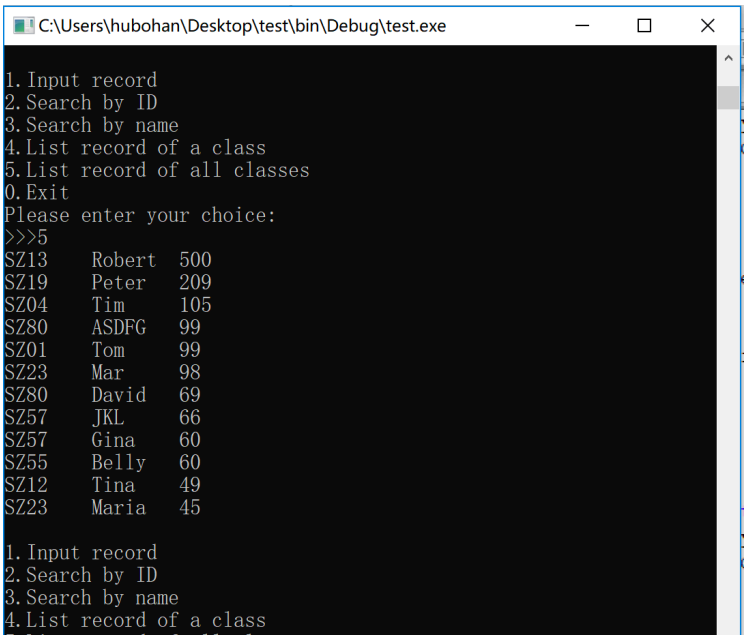

C:\Users\hubohan\Desktop\test\bin\Debug\test.exe
1.Input record
2.Search by ID
3.Search by name
4.List record of a class
5.List record of all classes
0.Exit
Please enter your choice:
>>>2
Input the ID:
SZ01
Class:1 SZ01 Tom 99


C:\Users\hubohan\Desktop\test\bin\Debug\test.exe
1.Input record
2.Search by ID
3.Search by name
4.List record of a class
5.List record of all classes
0.Exit
Please enter your choice:
>>>2
Input the ID:
AAAA
Not Found!

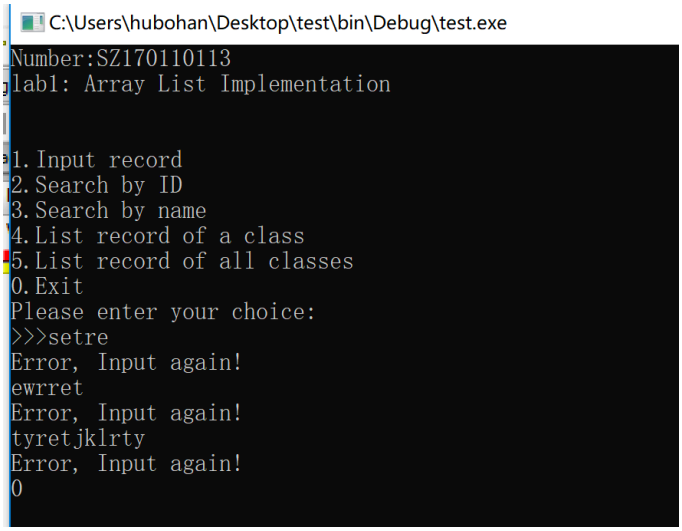

C:\Users\hubohan\Desktop\test\bin\Debug\test.exe
>>>1
Input the class:
4
Input the number:
SZ57
Input the name:
JKL
Input the score:
66
1.Input record
2.Search by ID
3.Search by name
4.List record of a class
5.List record of all classes
0.Exit
Please enter your choice:
>>>3
Input the name:
Tom
Class:1 SZ01 Tom 99


C:\Users\hubohan\Desktop\test\bin\Debug\test.exe
1. Input record
2. Search by ID
3. Search by name
4. List record of a class
5. List record of all classes
0. Exit
Please enter your choice:
>>>3
Input the name:
dskdlzddslf
Not Found!


C:\Users\hubohan\Desktop\test\bin\Debug\test.exe
2. Search by ID
3. Search by name
4. List record of a class
5. List record of all classes
0. Exit
Please enter your choice:
>>>4
Input the class:
1
Class:1
SZ04 Tim 105
SZ01 Tom 99
1. Input record
2. Search by ID
3. Search by name
4. List record of a class
5. List record of all classes
0. Exit
Please enter your choice:
>>>4
Input the class:
2
Class:2
SZ13 Robert 500
SZ19 Peter 209
SZ23 Mar 98
SZ80 David 69
SZ12 Tina 49
SZ23 Maria 45


C:\Users\hubohan\Desktop\test\bin\Debug\test.exe
1. Input record
2. Search by ID
3. Search by name
4. List record of a class
5. List record of all classes
0. Exit
Please enter your choice:
>>>5
SZ13 Robert 500
SZ19 Peter 209
SZ04 Tim 105
SZ80 ASDFG 99
SZ01 Tom 99
SZ23 Mar 98
SZ80 David 69
SZ57 JKL 66
SZ57 Gina 60
SZ55 Belly 60
SZ12 Tina 49
SZ23 Maria 45
1. Input record
2. Search by ID
3. Search by name
4. List record of a class

菜单入口处防御：



在需要输入整型变量的入口处做防御（班级、成绩）：