学生成绩管理系统的实现

教案

参赛组别：专业课程一组
所属课程：数据结构及算法设计
授课对象：计算机应用工程专业
参赛学时：16学时

虚实结合　　任务先行

目 录

《数据结构及算法设计》教案 1，3-4 学时

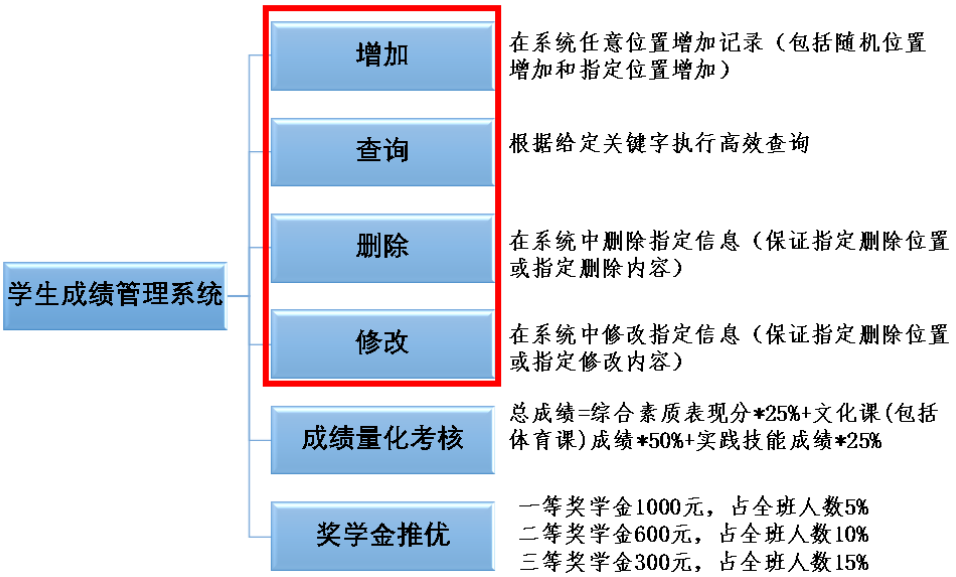
一、教学基本信息			
课程名称	线性表的顺序存储与实现	**授课教师**	于璐
授课班级	21 计算机应用工程 1 班	**授课时数**	2
授课时间	周五 5-6 节课	**授课地点**	81107

二、教学分析	
教学内容	1、线性表的概念、特点及其基本操作定义； 2、线性表的顺序存储逻辑及抽象数据类型； 3、线性表顺序存储的结构及特点； 4、分析顺序表的各种算法实现。
学情分析	本节课为第二单元线性表的第一个知识点。在此之前，学生已经学习了四大数据结构的特点以及一些基本的操作类型，并且具备了利用 C 语言进行较为复杂程序设计的实践技能，能够根据功能需求设计基本程序流程图并实现。因此，在本节课授课过程中，将重点对线性表的顺序存储逻辑结构和功能实现原理进行讲解和演示，学生在理解线性表运行原理的基础上，利用自身程序设计的技能，完成线性表顺序存储功能的设计和实现。

三、教学目标确定		
教学目标	**知识目标**	1、掌握线性表的定义及其逻辑结构； 2、理解线性表的抽象数据类型； 3、掌握顺序表的存储结构及特点； 4、掌握顺序表的基本运算：插入、删除及查找运算及其性能分析。
	能力目标	1、具有查阅资料、自主学习的能力； 2、具有初步算法分析和设计的能力； 3、具有独立学习，获取新知识和技能，在工作中发现问题、与分析问题、解决问题的能力。
	素质目标	1、具备依据实际问题的需求合理地组织数据，并在计算机中有效地存储数据的能力； 2、具备为解决实际问题进行算法设计与分析的能力； 3、具备将算法通过具体的编程语言加以实现的能力。
教学重点		1、线性表的类型定义及逻辑结构； 2、顺序表基本运算的实现及其性能分析。
教学难点		1、线性表与线性结构的联系与区别； 2、线性表的顺序存储结构及其运算。

四、思政融入课堂
通过讨论顺序结构的“优劣”，向学生传达唯物辩证法基本观点。

五、课程在项目中的定位

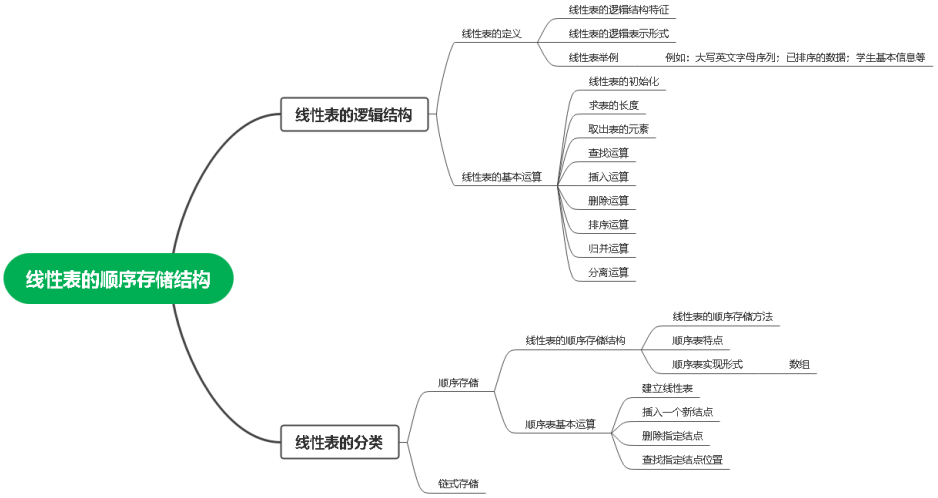


采用顺序存储结构生成顺序表，实现增、删、查、改四项基本功能。

六、教学策略

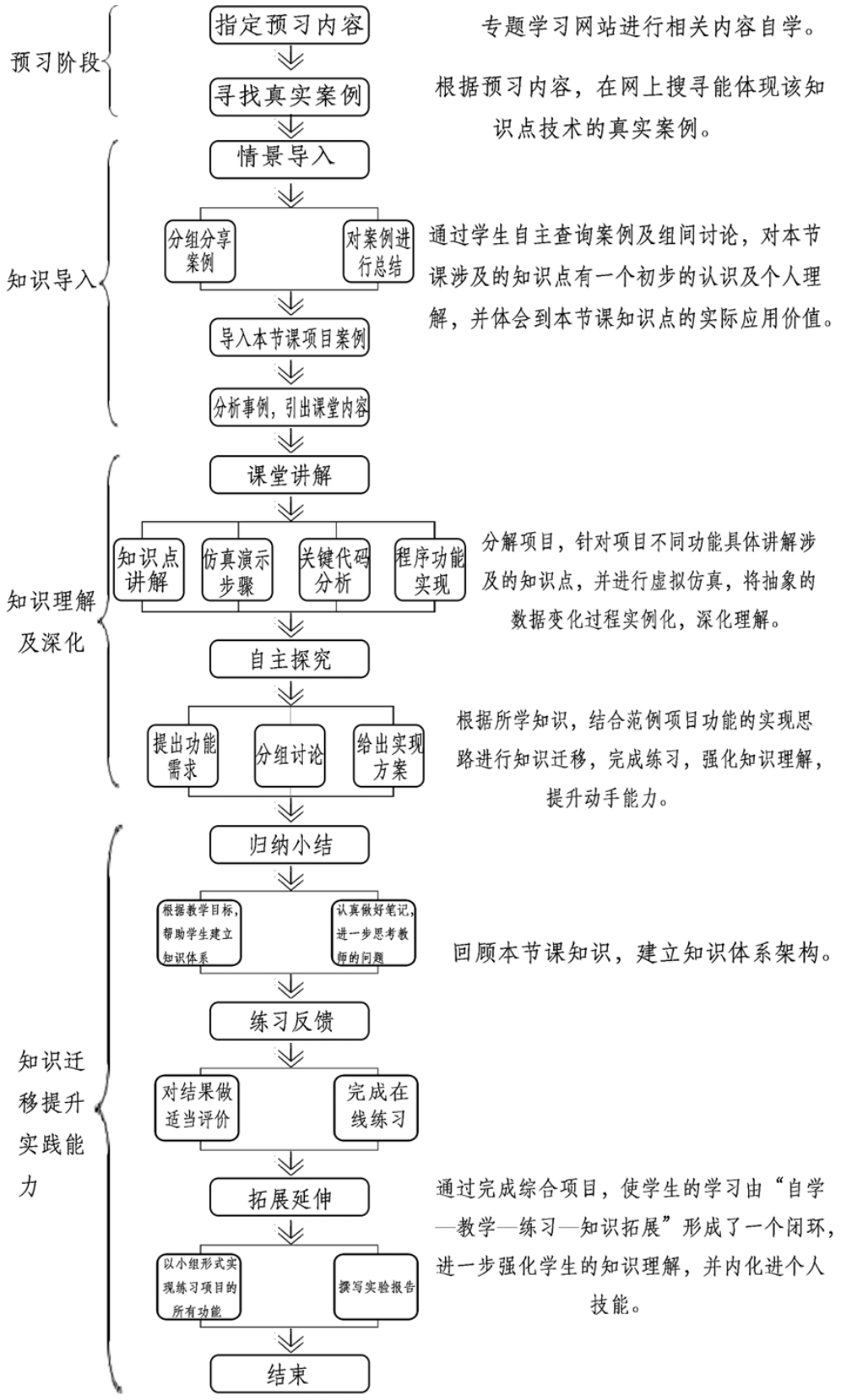
设计思路

1、 给出线性表的定义，注意强调要点，分析其逻辑结构；
2、 给出线性表的抽象数据类型；
3、 给出顺序表的存储结构示意图，强调存储要点，总结存储特点；
4、 根据存储特点，给出定义顺序表的结构体形式；
5、 利用算法动画演示，分析顺序表的插入运算执行过程，写出插入算法并分析；
6、 利用算法动画演示，分析顺序表的删除运算执行过程，写出删除算法并分析；
7、 利用算法动画演示，分析顺序表的查找运算执行过程，写出查找算法及分析。



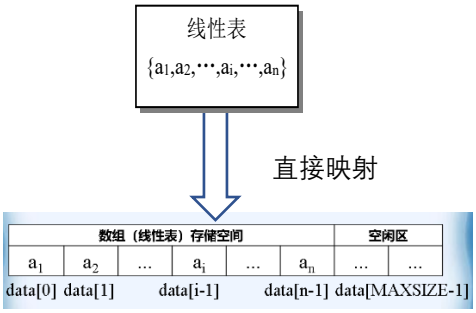
教学资源

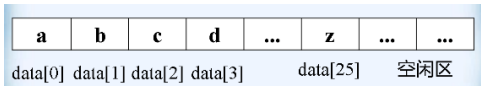
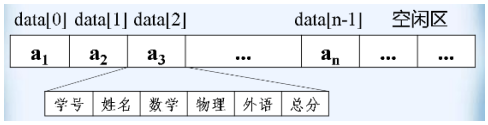
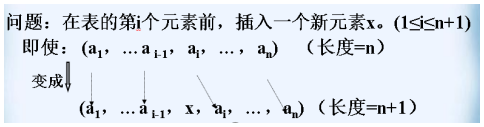
学习环境：机房、局域网，交互式电子黑板
学习资源：
1、 专题学习网站：包括慕课网、学习通、职教云等；
2、 授课课件：根据此节学习内容制作的 PPT 课件；
3、 多媒体资源库：虚拟仿真演示案例、项目演示案例；
4、 案例库：课堂练习题库、测验题库等；

	5、 精品课程网站：本门课程的精品课程网站； 8、 VC6.0++运行环境：进行案例演示及学生练习项目的专业运行环境。
教学流程	

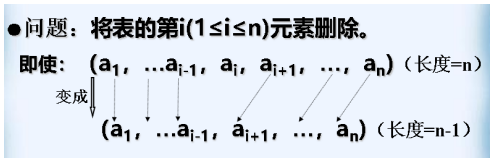
七、教学过程

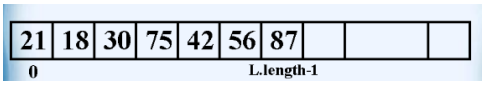
教学环节	教学活动		时间安排	设计思路与教学手段
	教师授课内容	学生		
课前预习				
1	1、提供专题学习网站上预习章节：线性表顺序存储结构； 2、思政学习：ChatGPT 模型 （1）https://zhuanlan.zhihu.com （2）/p/589621442	在线学习相关知识，分组网上寻找相关案例。	课前一周	**设计思路：** 1、利用专题网站自主学习； 2、根据个人对知识点的理解在网络进行信息查询，锻炼自主获取知识的能力。 **涉及教学资源：** 慕课网、信息搜索引擎
知识导入				
2	**分析学生案例**，探讨线性表在实际应用中的作用和意义。	以组为单位分享线性表的实际应用案例。	3min	**教学手段：** 分组讨论，案例总结。
3	**引出课堂案例**：有一组有序的数据{3，7，9，16，32，78，54}，现对该组数据进行相应处理，实现如下功能： 1、在该组数据中，插入数字 45，并保持数据的有序性； 2、删除第 3 个数据； 3、查询数据 16 在第几个位置。	观察案例，思考功能实现方法。	1min	**教学设计思路：** 选用贴近学生生活的案例来创设情境引入新课，让学生利用已有知识思考解决问题的方法。在后续课程学习中，通过算法对比，认识到利用线性表解决这些问题的优点和劣势，自主将课本知识内化为个人解决问题的经验。
知识讲解				
4	**板书重点知识：** **线性表的定义以及逻辑表达方法** **1、线性表定义：** 线性表是 n(≥0)个数据元素 a_1，a_2，…a_n的有限序列；表中每个元素（除第一个和最后一个外），有且仅有一个直接前趋，有且只有一个直接后继。即线性表或为一个空表（n=0），或为： （a_1，a_2，…，a_{i-1}，a_i，a_{i+1}，… ，a_n）（n>0）。	1、理解线性表的概念以及逻辑表达形式； 2、思考其结构特点； 3、对比之前课堂收	8min	**教学设计思路：** 1、理论讲解，了解线性表的概念和特点，通过案例分析将理论知识与实际应用间建立连接； 2、课堂讨论。通过讨论的形式对课前收集的案例进行分析，找出错误的案例，并说明其错误的地方。找对错的形式可以很

	2、线性表的逻辑表示形式： 线性表是一种非常典型的线性结构，用二元组可以表示成： S=（D，R） D={ a_1，a_2 ，…，a_i ，…，a_n} R={<a_1，a_2>，<a_2，a_3>，…，<a_i ，a_{i+1} >，…，<a_{n-1}，a_n >}。 **3、 举例说明：** （1）26 个英文字母组成的字母表； （2）一组有序数字； （3）学生成绩表。	集的案例，分析哪些是线性表结构，哪些不是； 4、拓展联系其他具有线性结构的数据实例。		好的帮助学生深化理论知识。 **教学资源：** PPT 讲解，案例分析。
5	**线性表的顺序存储结构** **1、线性表的顺序存储方法；** **2、什么是顺序表；** **3、顺序表特点；** **4、顺线性表的顺序存储结构可用数组来实现。** 假设用数组 data[MAXSIZE]来存储线性表 A=（a_1，a_2 ，…，a_i ，…，a_n），则线性表 A 对应的顺序存储结构为  **5、顺序表的寻址公式：** LOC(a_i)=LOC(a_1)+(i-1)*d **6、定义顺序表的结构类型：** # define MAXSIZE 1000 /* maxsize 称为顺序表的容量,表空间的大小可根据实际需要而定，这里假设为 1000*/ typedef int datatype; /*datatype 类型可根据实际情况而定，这里假设为 int */ typedef struct selist /*结构类型*/ { dataType data[MAXSIZE] ; /*存储空间*/ int last ; /*当前线性表的长度*/ } sequenlist; ; **7、举例：** 26 个英文字母的存储形式	1、了解线性表顺序存储结构的方法和实现方式； 2、掌握顺序存储结构中数据元素的寻址方式； 3、理解 26 个英文字母的存储形式； 4、思考学生成绩表在内存中的数据结构形式，尝试画出存储形式。	10min	**教学设计思路：** 1、理论讲解，以图形+文字的形式帮助学生理解顺序表存储结构； 2、知识迁移，利用前导课程“C 语言程序设计”中数组的知识，帮助学生理解数据在内存中的地址分配关系，顺序存储中元素寻址的过程，并总结寻址公示； 3、通过案例分析，以黑板画图的形式逐步展示 26 个英文字母在内存中的逻辑存储方式，深化学生对理论知识的理解； 4、知识拓展，学生利用所学知识，尝试设计学生成绩表的数据存储形式； 5、引导学生思考单个数据和多个数据在进行存储时，数据的逻辑结构的区别和联系，培养思考问题的全面性，拓展思维。 **教学资源：** PPT 讲解， 示例图演示，案例分析，拓展项目。

	8、知识拓展： 学生成绩统计表 学号 | 姓名 | 数学 | 物理 | 外语 | 总分 1 | 李华 | 88 | 89 | 90 | 267 2 | 王芳 | 98 | 90 | 87 | 275 3 | 张丽 | 78 | 84 | 90 | 252 4 | 田爽 | 89 | 69 | 78 | 236 （1）思考该组数据是否是线性表（巩固对线性表定义的理解）； （2）思考单体数据和复合式数据在进行逻辑存储时，其存储方式间的区别。			
6	**分析课堂练习，画图说明顺序表存储数据时，存储单体数据和复合体数据的异同** 1、单体数据，例如 26 个英文字母  2、复合数据，例如学生成绩表  学生成绩统计表的存储结构类型说明为： #define MAX 500 Typedef struct node {char no[10]; char name[10]; float score[5]; }datatype; typedef struct selist { dataType data[MAX] ; int last ; } sequenlist; ;	1、掌握不同数据的存储方式； 2、理解并掌握不同数据的存储结构的声明方式； 3、思考数组的空间占用情况以及不同数据形式的寻址方式。	5min	**教学设计思路：** 1、案例分析。通过不同数据内部存储形式的对比，强化学生的逻辑思维和空间想象能力； 2、理论讲解，利用前导课程“C 语言程序设计”中结构体的知识，理解复合式数据的类型定义方式，将程序代码与抽象的空间想象联系起来； 3、知识迁移。根据单数据的寻址方式，引导学生思考如果寻址复合数据，深化知识理解。 **教学资源：** PPT 讲解， 示例图演示，案例分析，拓展项目。
7	**顺序表的插入运算** **1、图例演示插入原理：**  **2、动画演示插入过程。** **3、课堂讨论总结插入步骤：** 将表中位置为 n ，n-1，…，i 上的结点，依次后移到位置 n+1，n，…，i+1 上，空出第 i 个位置 （1）在该位置上插入新结点 x。仅当插入	1、观察图例演示和动画演示，了解数据元素插入过程； 2、根据对插入过程的理解小组讨论插	10min	**教学设计思路：** 1、图例和动画演示可以帮助学生在脑海中构建插入的动态变化，建立直观感受； 2、课堂讨论过程中，学生将动态过程用语言进行描述，总结插入步骤，构建知识的逻辑框架； 3、课堂练习帮助进行知识拓展，帮助学生在理解

	位置 i=n+1 时，才无须移动结点，直接将 x 插入表的末尾； （2）该顺序表长度加 1。 注意： （1）当表空间已满，不可再做插入操作； （2）当插入位置为非法位置，不可做正常插入操作。 **4、知识拓展：**在 24 个有序英文字母中分别插入字母 a 和 z，口述插入过程。 **5、课堂讨论：**不同位置的插入在时间复杂度上是否区别，其程序运行效率是否有不同。	入步骤； 3、课堂回答问题，将所学知识进行实际应用，深化知识理解； 4、根据数据插入的步骤，思考在顺序存储中，插入位置不同对程序运行效率的影响，进行扩展思维练习。		知识点的基础上，全面考虑实际应用中可能出现的其他问题，拓展思维。 **教学资源：** PPT 讲解， 示例图演示，动画演示，拓展项目。
8	**项目实现，在有序数据{3，7，9，16，32，78，54}中插入数字 45，并保持数据的有序性** 1、根据数组的特性，引导学生画出这组数据在内存中存储的逻辑地址示意图； 2、对比数据存储示例图，口述插入数字 45 的步骤； 3、知识迁移，根据插入步骤，画出程序流程图； 4、根据程序流程图，进行相应程序的编程并实现； 5、知识拓展，尝试解决在 25 个有序英文字母中，插入字母 y 的过程及程序实现。	1、在练习册上画出数据组顺序存储的逻辑地址示意图； 2、回顾插入过程，思考如何将数据 45 插入到数据组中； 3、根据插入步骤绘制程序流程图； 4、结合程序流程图，理解程序段的含义； 5、课堂练习，实现英文字母的插入。	12min	**教学设计思路：** 1、课堂练习，手绘数据存储的地址结构，并口述数据插入过程，帮助学生巩固顺序存储的特点，巩固理论知识； 2、知识迁移，程序的实现帮助学生在逻辑结构和功能实现之间建立连接，将理论知识转移为实践能力； 3、知识拓展，根据现有实践经验进行知识拓展，解决这一类的问题，深化知识理解，提升实践技能。 **教学资源：** PPT 讲解， 示例图演示，上机编程并运行。

9	**顺序表的删除运算** **1、图例演示删除原理**  **2、动画演示删除过程** **3、课堂讨论总结删除步骤** （1）若 i=n，则只要删除终端结点，无须移动结点； （2）若 1≤i≤n-1，则必须将表中位置 i+1，i+2，…，n 的结点，依次前移到位置 i，i+1，…，n-1 上，以填补删除操作造成的空缺； （3）该表长度减 1。 注意： 当要删除元素的位置 i 不在表长范围内即 i＜1 或 i＞L->length）时，为非法位置，不能做正常的删除操作。 **4、知识拓展：**在 26 个有序英文字母中分别删除第一个字母和最后一个字母，口述删除过程。 **5、课堂讨论：**不同位置的删除在时间复杂度上是否区别，其程序运行效率是否有不同。	1、观察图例演示和动画演示，了解数据元素删除过程； 2、根据对删除过程的理解小组讨论删除步骤； 3、课堂回答问题，将所学知识进行实际应用，深化知识理解； 4、根据数据删除的步骤，思考在顺序存储中，删除位置不同对程序运行效率的影响，进行扩展思维练习。	9min	**教学设计思路：** 1、图例和动画演示可以帮助学生在脑海中构建删除的动态变化，建立直观感受； 2、课堂讨论过程中，学生将动态过程用语言进行描述，总结删除步骤，构建知识的逻辑框架； 3、课堂练习帮助进行知识拓展，帮助学生在理解知识点的基础上，全面考虑实际应用中可能出现的其他问题，拓展思维。 **教学资源：** PPT 讲解， 示例图演示，动画演示，拓展项目。
10	**项目实现，在有序数据{3，7，9，16，32，78，54}中删除第 3 个数据** 1、根据数组的特性，引导学生画出这组数据在内存中存储的逻辑地址示意图； 2、对比数据存储示例图，口述删除第 3 个数据的步骤； 3、知识迁移，根据删除步骤，画出程序流程图； 4、根据程序流程图，进行相应程序的编程并实现； 5、知识拓展，尝试解决在 26 个有序英文字母中，删除第 3 个字母的过程及程序实现。	1、在练习册上画出数据组顺序存储的逻辑地址示意图； 2、回顾删除过程，思考如何删除第 3 个数据； 3、根据删除步骤绘制程序流	8min	**教学设计思路：** 1、课堂练习，手绘数据存储的地址结构， 并口述数据删除过程，帮助学生巩固顺序存储的特点，巩固理论知识； 2、知识迁移，程序的实现帮助学生在逻辑结构和功能实现之间建立连接，将理论知识转移为实践能力； 3、知识拓展，根据现有实践经验进行知识拓展，解决这一类的问题，深化

		程图； 4、结合程序流程图，理解程序段的含义； 5、课堂练习，实现英文字母的删除。		知识理解，提升实践技能。 **教学资源：** PPT 讲解， 示例图演示，上机编程并运行。
11	**顺序表的查找运算** **1、图例演示查找原理**  **2、动画演示查找过程** **3、课堂讨论总结查找步骤** 从第一个元素 a1 起依次和 x 比较，直到找到一个与 x 相等的数据元素，则返回它在顺序表中的存储下标或序号；或者查遍整个表都没有找到与 x 相等的元素，返回 0。 **4、知识拓展：**在 26 个有序英文字母中查找字母 f，返回具体位置信息，口述查找过程。 **5、课堂讨论：**不同位置的查找在时间复杂度上是否区别，其程序运行效率是否有不同。	1、观察图例演示和动画演示，了解数据元素查找过程； 2、根据对查找过程的理解，小组讨论查找步骤； 3、课堂回答问题，将所学知识进行实际应用，深化知识理解； 4、根据数据查找的步骤，思考在顺序存储中，查找位置不同对程序运行效率的影响，进行扩展思维练习。	6min	**教学设计思路：** 1、图例和动画演示可以帮助学生在脑海中构建查找的动态变化，建立直观感受； 2、课堂讨论过程中，学生将动态过程用语言进行描述，总结查找步骤，构建知识的逻辑框架； 3、课堂练习帮助进行知识拓展，帮助学生在理解知识点的基础上，全面考虑实际应用中可能出现的其他问题，拓展思维。 **教学资源：** PPT 讲解， 示例图演示，动画演示，拓展项目。
12	**项目实现，在有序数据{3，7，9，16，32，78，54}中查询数据 16 在第几个位置** 1、根据数组的特性，引导学生画出这组数	1、在练习册上画出数据组顺	12min	**教学设计思路：** 1、课堂练习，手绘数据存储的地址结构， 并口

	据在内存中存储的逻辑地址示意图； 2、对比数据存储示例图，口述查找目标数据的步骤 ； 3、知识迁移，根据查找步骤，画出程序流程图； 4、根据程序流程图，进行相应程序的编程并实现； 5、知识拓展，尝试解决在 26 个有序英文字母中，查找字母 m 的过程及程序实现。	序存储的逻辑地址示意图； 2、回顾查找过程，思考如何删除目标数据； 3、根据查找步骤绘制程序流程图； 4、结合程序流程图，理解程序段的含义； 5、课堂练习，实现英文字母的查找。		述数据查找过程，帮助学生巩固顺序存储的特点，巩固理论知识； 2、知识迁移，程序的实现帮助学生在逻辑结构和功能实现之间建立连接，将理论知识转移为实践能力； 3、知识拓展，根据现有实践经验进行知识拓展，解决这一类的问题，深化知识理解，提升实践技能。 **教学资源：** PPT 讲解， 示例图演示，上机编程并运行。
课堂小结				
13	线性表顺序存储的优缺点 **1、特点：**在逻辑上相邻的元素在物理上也相邻，即顺序存储是用物理位置上的邻接关系来表示元素间的逻辑关系。 **2、优点：**结构简单，存取方便、迅速。 **3、缺点：** （1）作插入、删除运算需移动大量数据； （2）表难以扩充（因需一片连续空间）。 **思政学习：**顺序结构的“优劣”，向学生传达唯物辩证法基本观点。	回顾本节课知识，构建知识架构。	2min	**教学设计思路：** 1、便于构成了一个完整的知识体系。使学生在自己的头脑中形成知识网络．从而达到提纲挈领的目的； 2、使学生带着问题进入教室，带着更多的问题离开教室，培养学生探究精神。
项目分析				
14	课堂连线企业导师，对本节课利用顺序表表实现系统功能进行项目分析，说明评价标准，强调注意事项。	根据企业导师点评进行自我反思，及时进行自我学习调整，分析本节课项	4min	**教学设计思路：** 企业导师根据行业标准进行项目分析，帮助学生捋清解决问题的思路和注意事项，强化对知识的理解，提升学生的职业素养。

		目实现的设计要点，进行算法设计。		

八、知识拓展

1、拓展训练

学生成绩统计表

学号	姓名	数学	物理	外语	总分
1	李华	88	89	90	267
2	王芳	98	90	87	275
3	张丽	78	84	90	252
4	田爽	89	69	78	236

用程序实现以下功能：

（1）插入新的一条记录“孟想，79，76，95，250”；

（2）删除第 3 条记录；

（3）查询王芳在第几条记录；

（4）依次输出表中所有记录。

2、考证（考研）训练

（2013 年考研真题）已知两个长度分别为 m 和 n 的升序顺序表，若将它们合并为一个长度为 m+n 的降序顺序表，则最坏情况下的时间复杂度是（）。

A. O（n） B. O（mn） C. O（min(m, n)） D. O（max(m, n)）

（软件水平考试）在一个长度为 n 的顺序储存的线性表中，删除地 i 个元素(1≤i≤n)时，需要从前向后依次移（ ）个元素。

A. I B. n-i+1 C. n-i-1 D. n-i

3、竞赛训练

蓝桥杯线性表部分 https://dasai.lanqiao.cn/notices/1096/

九、教学评价与反馈

本节课是数据结构第一种类型—线性结构的第一节课，涉及到线性结构的特征、在内存中的存储形式、以及在进行数据操作过程中内存地址的变化过程。理论知识较为抽象，需要学生加入个人的空间想象辅助理解。此外，在建立了空间逻辑结构的基础上还需要将其以程序的形式进行实现，因此，课程的实操性也很强。

在整个教学过程中，通过前期的课堂预习，学生在课前已经对线性结构有了初步的了解，大部分学生能够给依据对知识的理解找到对应的生活实例，在理论与现实生活间建立了联系，认识到所学内容所具备的实际意义。

在课堂教学中，对于理论性较强的部分，通过导学案充分铺垫、诱导、启发、示范、练习，以及教学支架性材料，讲解时以例释理，辅以示例图和关键步骤的动画演示，以自主总结归纳替代直接灌输结论，发挥学生的主观能动性进行知识点的学习、深化，构建属于自己的知识体系架构，极大的提升了学生上课的注意力，对知识的理解也较为透彻。而对于实践技能训练部分，通过小组讨论总结知识点、课堂回答问题、课堂练习等多种形式相结合的形式，每个学生都充分、全程并且深度参与课堂互动中，在体验、经验的基础上领悟、归纳、总结知识

点，通过理解程序——仿写程序——知识迁移的形式逐步将抽象的数据结构形式转化为个人对程序的理解，并实现了其功能。达到了预想的教学效果。

在课后知识拓展部分，对线性结构的顺序存储应用进行了拓展，将单一数据的增删查拓展为复合数据的增删查。需要用到前导课程“C 语言程序设计”中结构体部分的知识内容。虽然在课堂上对于复合数据的逻辑存储形式进行了讲解和练习，大部分学生也能明确的阐述这样的数据结构在增删查操作中与单一数据操作的区别和联系，但是在进行实际程序实现过程中，仍出现无法通过程序将复合数据转换为线性结构进行顺序存储。出现这样的情况主要涉及到两方面问题：

1、前导课程“C 语言程序设计”中结构体部分的知识内容没有完全掌握，无法做到将结构体应用到数组的存储应用中；

2、对于线性表顺序存储结构的增、删、查的逻辑变化没有完全理解，尤其在数据操作过程中其地址的变化过程和指针的行为没有完全理解。

在后续的学习中，应该对前导课程“C 语言程序设计”指针和结构体两部分知识进行强化练习，提升基本程序的操作技能，此外，通过学习复合数据存储的范例程序，尝试解决类似的其他复合数据的相关功能，完成知识的拓展，深化理论知识理解。

十、教学总结

本节课采用项目教学法展开教学，通过学生自己在学习和生活经历中寻找到的线性表结构实例对线性存储结构有了初步的认知，课堂上通过一个简单的有序数据的增、删、查的基本操作，了解到在内存中数据的地址变换方式，将抽象的空间想象以具体的程序进行了实现，建立逻辑结构与功能实践之间的联系，深化理论知识理解，提升实践技能。

在本节课中，理论知识涉及到内存地址的变化，较为抽象，在讲解过程中采用“示例图讲解+动画演示+学生自主总结”的方式进行理论知识讲解，有效的将课堂知识内化到学生自身构建的知识体系中。实践操作部分，涉及到用程序实现内存地址的变化过程，对学生的编程能力要求较高，因此，在讲解过程中，采用“构建动态变化的直观印象+流程图总结步骤+程序仿写+知识拓展”的形式，帮助学生建立抽象逻辑结构与程序算法之间的联系，在仿写程序过程中一方面进一步熟悉程序的算法思想，另一方面深化理论知识理解，为后续的知识迁移和拓展做准备。

在整个课堂实施过程中，通过课堂回答问题、课堂练习以及小组讨论的形式，使学生全程参与课堂教学始终，保持了良好的、积极的课堂氛围。

十一、项目实现代码

```
#include <stdio.h>
#include <string.h>
struct data
{
    int number;
    char name[10];
    char sex[5];
    char classes[5];
   int tel;
};
int insert(struct data st[5],int n); /*申明插入函数*/
void scanning(struct data st[5],int n); /*申明浏览函数*/
int del(struct data st[5],int n); /*申明删除函数*/
```

```
void fix(struct data st[5],int n); /*申明修改函数*/
int main()
{
    struct data st[5];
    int chioce,flag=1;
    int n;
    n=0;
    while(flag)
    {
        printf("请选择功能:\n          1—信息浏览\n          2—插入信息\n          3—删除信息\n
4-修改信息\n          0—退出程序\n");
        scanf("%d",&chioce);
        switch(chioce)
        {
            case 1:
                scanning(st,n);
                break;
         case 2:
                n=insert(st,n);
                break;
         case 3:
                n=del(st,n);
                break;
             case 4:
                fix(st,n);
                break;
              case 0:
                  flag=0;
                break;
       }
    }
    printf("\n");
    printf("谢谢使用！\n");
    return 0;
    }
void scanning(struct data st[5],int n)    /*定义浏览函数*/
{
    int i;
    if(n==0) /*无元素*/
    {        printf("请选择功能键，先插入名单！\n");
        printf("\n");
                printf("\n");
    }
    else
    {
```

```
        for(i=1;i<=n;i++)
        {
        printf("    学号：%d\n ",st[i].number);
        printf("    姓名：%s\n",st[i].name);
        printf("    性别：%s\n",st[i].sex);
        printf("    班级：%s\n",st[i].classes);
        printf("联系方式：%d\n",st[i].tel);
        printf("\n");
        }
    }
}
int  insert(struct data st[5],int n)  /*定义插入函数*/
{   int i,p;
    struct data t;
    if(n>=5)
    {       printf("内存已满！\n");
            printf("\n");
    }
    else
    {
        printf("请输入学号：\n");
        scanf("%d",&t.number);
        printf("请输入姓名：\n");
        scanf("%s",t.name);
        printf("请输入性别：\n");
        scanf("%s",t.sex);
        printf("请输入班级：\n");
        scanf("%s",t.classes);
        printf("请输入号码：\n");
        scanf("%d",&t.tel);
    }
    for(i=n;i>0;i--) /*所有元素后移*/
    {
        st[i+1].number=st[n].number;
        strcpy(st[i+1].classes,st[i].classes);
        strcpy(st[i+1].name,st[i].name);
        strcpy(st[i+1].sex,st[i].sex);
         st[i+1].tel=st[i].tel;
   }
        st[1].number=t.number;
       strcpy(st[1].classes,t.classes);
       strcpy(st[1].name,t.name);
       strcpy(st[1].sex,t.sex);
       st[1].tel=t.tel;
    p=n+1;
```

```
    return p;
}
int del(struct data st[5],int n) /*定义删除函数*/
{
    int m,i,q,t;
    printf("请输入你要删除的同学的学号：\n");
    scanf("%d",&m);
    for(i=1;i<=n;i++)
    {       if(st[i].number==m)
        {
            t=i;
            break;
        }   }
    for(i=t;i<n;i++)   /*删除结点后的所有元素后移*/
    {
        st[i].number=st[i+1].number;
        st[i].tel=st[i+1].tel;
        strcpy(st[i].classes,st[i+1].classes);
        strcpy(st[i].name,st[i+1].name);
        strcpy(st[i].sex,st[i+1].sex);
    }
    q=n-1;
    return q;
}
void fix(struct data st[5],int n)  /*定义修改函数*/
{   int i,m,p,t=1;
    struct data s;
    printf("请输入你要修改的学生的学号：\n");
   scanf("%d",&m);
    for(i=1;i<=n;i++)
    {if(st[i].number==m)
          {while(t)
             {
             printf("你想要修改哪项数据？\n 1代表学号\n 2代表姓名\n 3代表性别\n 4代表班级\n 5代表联系方式\n
(注意：修改完毕请输入)\n");
         scanf("%d",&p);
         switch(p)
         {
         case 1:
                  printf("请输入修改后的学号！\n");
                  scanf("%d",&s.number);
                  st[i].number=s.number;
                  break;
         case 2:
                  printf("请输入修改后的姓名！\n");
```

```
            scanf("%s",s.name);
            strcpy(st[i].name,s.name);
            break;
        case 3:
            printf("请输入修改的性别！\n");
            scanf("%s",s.sex);
            strcpy(st[i].sex,s.sex);
            break;
        case 4:
            printf("请输入修改的班级！\n");
            scanf("%s",s.classes);
            strcpy(st[i].classes,s.classes);
            break;
        case 5:
            printf("请输入修改后的联系方式！\n");
            scanf("%d",&s.tel);
            st[i].tel=s.tel;
            break;
        case 0:
            t=0;
            break;          }    }    }    }}
```

《数据结构及算法设计》教案 2，5-6 学时

一、教学基本信息			
课程名称	线性表的链式存储与实现	**授课教师**	于璐
授课班级	21 计算机应用工程 1 班	**授课时数**	2
授课时间	周一 1-2 节课	**授课地点**	81107

二、教学分析	
教学内容	1、线性链表、单链表、静态链表的概念、表示及实现方法； 2、循环链表的概念； 3、双向链表的表示与实现； 4、分析顺序表与链表的差异。
学情分析	“链表”是学生接触的线性结构中的第二种存储形式。在前面的教学中，学生对于数据结构的概念应该已经比较清晰，也明确了顺序存储结构的组织形式及其应用，掌握了这一类数据结构的基本操作原理，对线性结构有了初步的认识和直观印象。因此，本节课将重点讲解链式存储结构的概念、表示及实现方法。通过对同一个项目分别进行顺序存储和链式存储的对比，学生会更加深刻的理解数组与链表这两种数据存储形式，并认识到不同的存储形式的适用范围。此外，学生在掌握了单链表相关操作的基础上，通过知识拓展，了解链表的其他结构形式，拓宽学生知识面，可以根据用户对功能的需求侧重点选择合理的数据结构编程实现、解决问题。

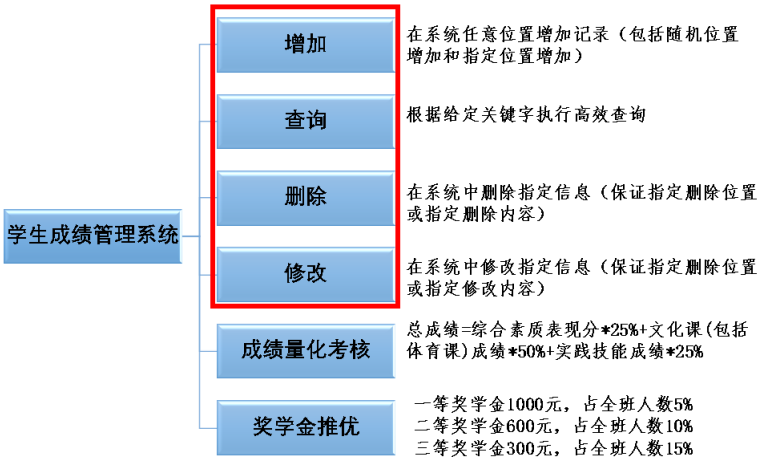
三、教学目标确定		
教学目标	**知识目标**	1、掌握单链表的结点结构； 2、掌握头结点、头指针和首元结点的概念； 3、单链表中数据关系的表示方法； 4、掌握线性表不同链式存储结构下基本操作的实现； 5、掌握链表和顺序表之间的异同。
	能力目标	1、具有查阅资料、自主学习的能力； 2、具有初步算法分析和设计的能力； 3、具有独立学习，获取新知识和技能，在工作中发现问题、与分析问题、解决问题的能力。
	素质目标	1、信息意识方面：使学生能够运用生活中的实例描述数据的内涵与外延，能够将有限制条件的、复杂生活情境中的关系进行抽象，用合理的数据结构表达数据的逻辑关系； 2、计算思维方面:使学生能够从数据结构的视角审视基于数组、链表的程序，解释程序中数据的组织形式，描述数据的逻辑结构及其操作，评判其中数据结构运用的合理性；能够针对限定条件的实际问题进行数据抽象，运用数据结构合理组织、存储数据，选择合适的算法编程实现、解决问题。
教学重点		1、链表的逻辑结构； 2、单链表的存储结构； 3、单链表的基本运算及其实现；

	4、双链表的逻辑结构与基本操作。
教学难点	1、单链表上实现的基本运算； 2、能理解数组、链表的区别，并能用程序实现链表的基本操作。选择合理的数据结构编程实现、解决问题。

四、思政融入课堂

强调在上机调试程序过程中，需要学生从一个又一个的错误中走出来，塑造学生克服困难，勇攀高峰的无畏精神。

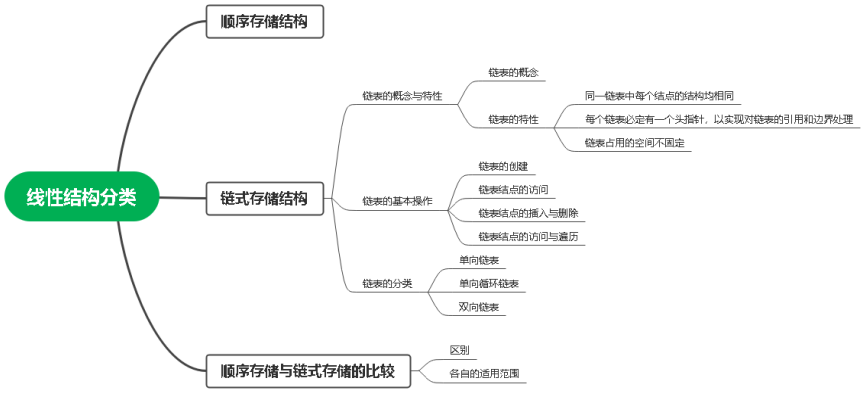
五、课程在项目中的定位



采用链式存储结构实现增、删、查、改四项基本功能，对比顺序表和链表的算法复杂度，通过不同的数据结构对程序进行优化，提升系统运行效率。

六、教学策略

教学资源	**学习环境**：机房、局域网，交互式电子黑板 **学习资源**： 6、 专题学习网站：包括慕课网、学习通、职教云等； 7、 授课课件：根据此节学习内容制作的 PPT 课件； 8、 多媒体资源库：虚拟仿真演示案例、项目演示案例； 9、 案例库：课堂练习题库、测验题库等； 10、 精品课程网站：本门课程的精品课程网站； 11、 VC6.0++运行环境：进行案例演示及学生练习项目的专业运行环境。
设计思路	1、给出单链表的存储结构示意图，强调存储要点，总结存储特点； 2、利用算法动画演示，分析单链表的插入、删除、查找运算执行过程，写出其算法并分析； 3、利用算法动画演示，分析单链表的头插法、尾插法建表运算执行过程，写出算法并分析； 4、给出循环链表的存储结构示意图，强调存储要点，总结存储特点； 5、给出双链表的存储结构示意图，强调存储要点，总结存储特点； 6、利用算法动画演示，分析双链表的插入、删除运算执行过程，写出查找算法及分析； 7、对 3 种不同结构的链表进行分析总结； 8、总结线性结构的顺序存储与链式存储的优缺点，分析其使用范围。



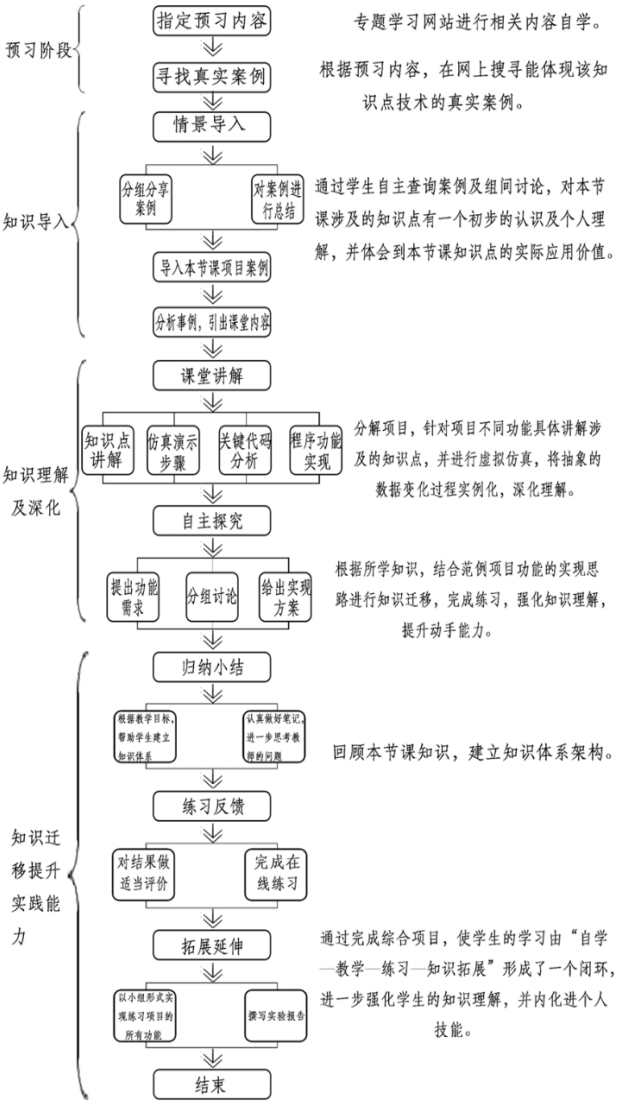
教学方法：

1、学生利用精品课及微课资源复习本节课知识，进行知识巩固；

2、教师通过学习通发布作业通知，学生接收作业，并根据所学知识选用适当的数据结构及算法进行任务实现；

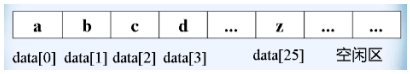
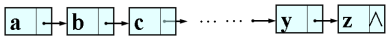
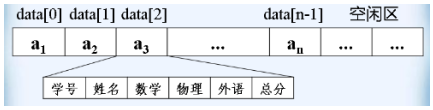
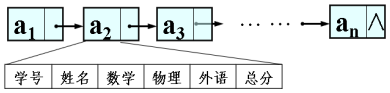
3、学生在线提交作业，教师根据学生提交的作业评估学生对于知识点的掌握情况，根据具体情况单独或集体进行答疑，并及时调整授课计划及授课内容。

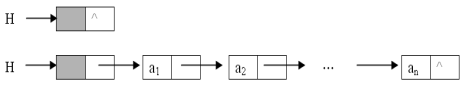
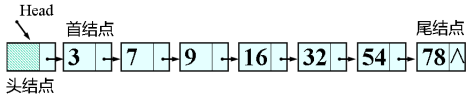
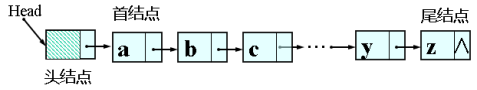
教学流程

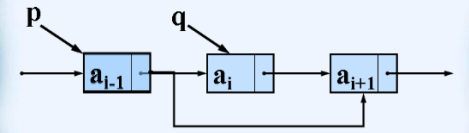


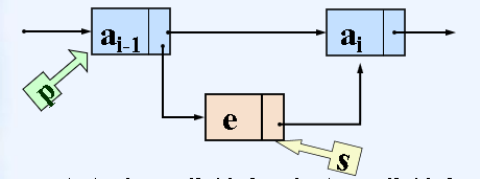
七、教学过程

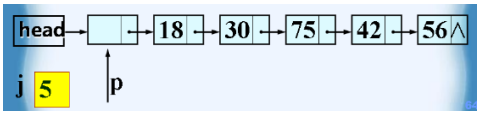
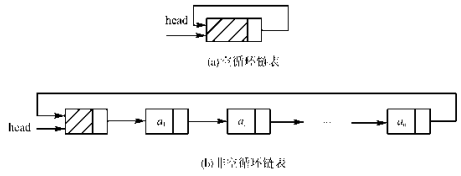
教学环节	教学活动		时间安排	设计思路与教学手段
	教师授课内容	学生		
课前预习				
1	1、提供专题学习网站上预习章节：线性表链式存储结构； 2、思政学习：AI 医疗影像行业发展； 3、问：顺序存储和链式存储哪种形式更方便、效率更高。	1、在线学习相关知识； 2、讨论顺序存储和链式存储哪种形式更方便、效率更高。	课前一周	**设计思路：** 1、利用专题网站自主学习； 2、根据个人对知识点的理解在网络进行信息查询，锻炼自主获取知识的能力。 **涉及教学资源：** 慕课网、信息搜索引擎。
知识导入				
2	**课程回顾** 1、顺序存储的存储和实现方式； 2、插入和删除元素操作需要移动大量的元素； 3、频繁增、删数据导致数据规模不稳，形成存储空间“碎片”； 4、限定最大空间，造成资源浪费。	以组为单位讨论顺序存储的局限性，尝试寻找解决问题的方法。	3min	**教学设计思路：** 排队与插队的生活案例，引导学生思考顺序数据结构在特定场景下的应用局限性，为链式存储结构的介绍做好铺垫。同时通过“数组”的过渡有利于加深学生对顺序存储模式和链式存储模式的理解。 **教学手段：** 分组讨论，案例分析。
3	**引出课堂案例：**有一组有序的数据{3，7，9，16，32，54，78}，现对该组数据进行相应处理，实现如下功能： 1、在该组数据中，插入数字 45，并保持数据的有序性； 2、删除第 3 个数据； 3、查询数据 16 在第几个位置。	观察案例，思考功能实现方法。	1min	**教学设计思路：** 选用贴近学生生活的案例来创设情境引入新课，让学生利用已有知识思考解决问题的方法。在后续课程学习中，通过对比认识到利用线性表和链表解决这些问题的优点和劣势，自主将课本知识内化为个人解

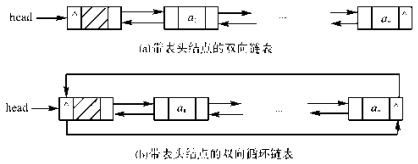
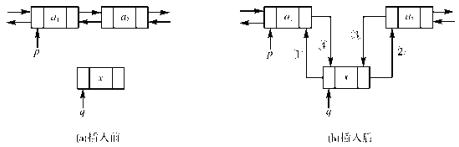
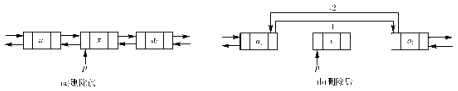
				决问题的经验。
知识讲解				
4	**板书重要知识点** **单链表的定义和存储形式** **1、动画演示链表在内存中的地址特点；** **2、单链表的结构特点：** 单链表—是由若干个结点组成，每个结点含两部分：数据域 data 和指针域 next；数据域 data 存放数据元素的值，指针域 next 存放下一个结点（直接后继）在存储器中的地址。n 个结点链结成一个链表，形成线性表的链式存储结构。 **3、讨论总结链表定义：** 用链接方式存储的线性表简称为链表 Link List。 链表的具体存储表示为： （1）用一组任意的存储单元来存放； （2）链表中结点的逻辑次序和物理次序不一定相同。还必须存储指示其后继结点的地址信息。 **4、对比顺序存储和链式存储的地址关系：** 26 个英文字母组成的字母表 顺序存储：  链式存储  **5、课堂练习** 画出学生成绩表的顺序存储和链式存储 顺序存储：  链式存储（每条记录是一个结点）：  	1、通过观看动画，了解单链表结点在内存中的分布特点以及组成线性结构的原理与关键技术点； 2、通过建立单链表的直观印象，总结单链表的定义和表示形式； 3、思考其结构特点； 4、拓展联系其他具有线性结构的复合数据实例。	8min	**教学设计思路：** 由于本节课涉及到内存地址的变化，较为抽象，因此，该部分内容主要以虚拟仿真的形式进行讲解。用动画结合实例进行演示，帮助学生建立内存变化的直观印象。包括： 1、仿真演示。采用上节课讲述顺序存储结构的案例，对比顺序存储和链式存储中，数据在内存中存储方式的不同，帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、课堂讨论。通过学生对数据地址动态变换过程的思考，总结链表的定义以及表现形式。在抽象的空间想象与实际案例之间建立连接； 3、知识拓展。从单一数据的链式存储拓展为复合数据的链式存储，进一步强化学生对于线性结构在不同应用中，其链式存储表达形式的敏感性。 **教学资源：** PPT 讲解，案例分析，虚拟仿真互动。
5	**单链表的结构及类型定义** **1、单链表的数据结构**	1、了解单链表结点的数据结构，在逻辑结构与程序实现之间建立连	5min	**教学设计思路：** 1、理论讲解，以图形+文字的形式 细化单链表结点的数据结构，完成从“逻

	 typedef struct node { datatype data ; struct node *next ; } ListNode ; ListNode *p; 结点分量的访问： 指针变量 p 的值——结点地址； 结点变量*p 的值——结点内容 ； p->data 的值——p 指针所指结点的 data 域的值 ； p->next 的值——结点 p 的后继结点的地址； **2、单链表的一般表示方式**  单链表特点： 起始节点又称为首结点，无前驱，故设头指针 head 指向开始结点。 链表由头指针唯一确定，单链表可以用头指针的名字来命名。头指针名是 head 的链表可称为表 head。 终端结点又称尾结点，无后继，故终端结点的指针域为空，即 NULL 除头结点之外的结点为表结点 为运算操作方便，头结点中不存数据； **3、项目实现** 有序数列{3，7，9，16，32，54，78}  ； **4、知识拓展** 26 个英文字母组成的字母表  。	接； 2、掌握单链表的一般表示方法，根据示例图总结单链表的特点； 3、规范有数序列的单链表表示形式； 4、尝试画出 26 个英文字母的单链表表示形式。		辑构建-空间构建-程序实践”的知识架构过程，学生在后续使用单链表进行增、删、改、查等相关操作时，更容易将动态的地址变化过程利用程序进行实现； 2、知识拓展，通过对单链表的一般表示形式的讲解，让学生了解到行业中类似数据的通用处理方法，拓展学生思路，让学生的实践技能更贴合行业技术要求； 3、分组讨论，总结单链表特点，加深对知识的理解，构建属于自身的知识架构； 4、深化练习：通过老师示范和学生自己动手练习相结合的模式，加强学生对于该知识点的理解与应用。 **教学资源：** PPT 讲解， 示例图演示，案例分析，拓展项目。
6	**链表的删除运算** **1、虚拟仿真演示删除原理；** **2、课堂讨论总结删除步骤：** 在单链表中删除第 i 个结点的基本操作为:找到线性表中第 i-1 个结点，修改其指向后继的指针； **3、图例演示，**回顾删除过程，辅以实现	1、观察仿真演示，了解数据元素删除原理； 2、根据对删除过程的理解小组讨论删除	6min	**教学设计思路：** 1、仿真演示。通过对结点动态变化过程的演示，帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程；

	的程序代码：  q = p->next; p->next = q->next; free(q); **4、课堂练习**，在 26 个有序英文字母中分别删除字母 a 和 z，口述删除过程； **5、思考**，不同位置的删除在时间复杂度上是否区别，其程序运行效率是否有不同。	步骤； 3、根据图例，利用前导课程“C 语言程序设计”中的“指针”知识点，用程序段描述地址变化过程； 4、课堂回答问题，将所学知识进行实际应用，深化知识理解； 5、思考删除位置不同对程序运行效率的影响，进行扩展思维练习。		2、分组讨论，帮助学生在讨论总结的过程中内化知识点，建立自己的知识架构； 3、图例演示：通过对删除步骤的分解，引导学生用程序语句进行描述，在逻辑抽象和实际实践之间建立连接； 4、思维拓展。通过对一些极端情况的讨论，思考其运行效率，培养学生在解决问题时进行多方面思考的能力和习惯。 **教学资源：** PPT 讲解，虚拟仿真，示例图演示，案例分析 。
7	**项目实现，在有序数据{3，7，9，16，32，78，54}中删除第 3 个数据** 1、根据链式存储的特性，引导学生画出这组数据在内存中存储的逻辑地址示意图； 2、根据数据存储示例图，口述删除第 3 个数据的步骤； 3、知识迁移，根据删除步骤，画出程序流程图； 4、根据程序流程图，进行相应程序的编程并实现； 5、知识拓展，尝试解决在 26 个有序英文字母中，删除字母 a 的过程及程序实现。	1、在练习册上画出数据链式存储的逻辑地址示意图； 2、回顾删除过程，思考如何删除第 3 个数据； 3、根据删除步骤绘制程序流程图； 4、结合程序流程图，理解程序段的含义； 5、课堂练习，实现英文字母的删除。	12min	**教学设计思路：** 1、课堂练习，手绘数据存储的地址结构，并口述数据删除过程，帮助学生巩固链式存储的特点，巩固理论知识； 2、知识迁移，程序的实现帮助学生在逻辑结构和功能实现之间建立连接，将理论知识转移为实践能力； 3、知识拓展，根据现有实践经验进行知识拓展，解决这一类的问题，深化知识理解，提升实践技能。 **教学资源：** PPT 讲解， 示例图演示，上机编程并运行。
8	**链表的插入运算** **1、虚拟仿真演示插入原理；** **2、课堂讨论总结插入步骤；** 具体步骤：	1、观察图例演示和动画演示，了解数据元素插入过	6min	**教学设计思路：** 1、仿真演示。通过对结点动态变化过程的演示，帮助学生建立

	（1）找到 ai-1 存储位置 p; （2）生成一个数据域为 x 的新结点*s; （3）令结点*p 的指针域指向新结点; （4）新结点的指针域指向结点 ai。 **3、图例演示**，回顾插入过程，辅以实现的程序代码：  s->next = p->next; p->next = s ; **4、课堂练习**，在 24 个有序英文字母中分别插入字母 a 和 z，口述插入过程; **5、思考**，不同位置的插入在时间复杂度上是否区别，其程序运行效率是否有不同。	程; 2、根据对插入过程的理解小组讨论插入步骤; 3、根据图例，利用前导课程“C 语言程序设计”中的“指针”知识点，用程序段描述地址变化过程; 4、课堂回答问题，将所学知识进行实际应用，深化知识理解; 5、思考插入位置不同对程序运行效率的影响，进行扩展思维练习。		抽象的空间想象，直观感受到地址的动态变换过程; 2、分组讨论，帮助学生在讨论总结的过程中内化知识点，建立自己的知识架构; 3、图例演示。通过对插入步骤的分解，引导学生用程序语句进行描述，在逻辑抽象和实际实践之间建立连接; 4、思维拓展。通过对一些极端情况的讨论，思考其运行效率，培养学生在解决问题时进行多方面思考的能力和习惯。 **教学资源：** PPT 讲解，虚拟仿真，示例图演示，案例分析 。
9	**项目实现，在有序数列{3，7，9，16，32，54，78}中插入数字 45，并保持数据的有序性** 1、根据链式存储的特性，引导学生画出这组数据在内存中存储的逻辑地址示意图; 2、根据数据存储示例图，口述插入数字 45 的步骤; 3、知识迁移，根据步骤画出程序流程图; 4、根据程序流程图，进行相应程序的编程并实现; 5、知识拓展，利用头插法或者尾插法创建单链表。	1、在练习册上画出数据链式存储的逻辑地址示意图; 2、回顾插入过程，思考如何插入 45 并保持数据有序性; 3、根据插入步骤绘制程序流程图; 4、结合程序流程图，理解程序段的含义; 5、课堂练习，实现单链表的建立。	12min	**教学设计思路：** 1、课堂练习，手绘数据存储的地址结构，并口述数据插入过程，帮助学生巩固链式存储的特点，巩固理论知识; 2、知识迁移，程序的实现帮助学生在逻辑结构和功能实现之间建立连接，将理论知识转移为实践能力; 3、知识拓展，根据现有实践经验进行知识拓展，解决这一类的问题，深化知识理解，提升实践技能。 **教学资源：** PPT 讲解， 示例图演示，上机编程并运行。

10	**链表的查找运算** **1、虚拟仿真演示查找原理；** **2、课堂讨论总结查找步骤：** （1）令计数器 j 为 0； （2）令 p 指向头结点； （3）当下一个结点不空时，并且 j<i 时， j 加 1，p 指向下一个结点； （4）如果 j 等于 i，则 p 所指结点为要找的第 i 结点，否则，链表中无第 i 结点。 **3、图例演示**，回顾插入过程，辅以实现的程序代码：  **4、项目实现**，在有序数列{3，7，9，16，32，54，78}查询数据 16 的位置 回顾前两个功能实现步骤，引导学生总结查询步骤，并细化不同查询情况的具体操作，包括定位查询和定值查询。 根据项目功能选择确认查询类型，绘制流程图并进行程序实现。 **5、课堂练习：** 对项目进行讨论，绘制另一种查询的流程图并进行程序实现。 **6、阶段总结：** 引导学生回顾课堂知识，讨论总结单链表的操作类型，以及各操作类型直接的区别和练习。	1、观察图例演示和动画演示，了解数据元素查找过程； 2、根据对查找过程的理解小组讨论插入步骤； 3、根据图例，利用前导课程“C 语言程序设计”中的“指针”知识点，用程序段描述地址变化过程； 4、根据现有知识积累，总结查询的步骤，细化查询的种类； 5、回顾之前所学，对单链表进行总结，分析各操作之间的区别和联系。	12min	**教学设计思路：** 1、仿真演示。帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、分组讨论。由于查询思想在之前删除和插入的程序算法中已有所涉及，因此，分组讨论可以提升学生的总结能力以及对之前的知识关键信息提取能力； 3、图例演示。引导学生用程序语句进行描述，在逻辑抽象和实际实践之间建立连接； 4、根据对插入和删除过程步骤的理解，提取出查询的程序流程图，并自行完成项目实现，内化理论知识，提升实践技能。 **教学资源：** PPT 讲解，虚拟仿真，示例图演示，案例分析。
11	**知识迁移及拓展** **链表的其他形式** **1、循环链表** （1）虚拟仿真演示循环链表的结构形式；  （2）与单链表对比，分析单链表和单循环链表之间的异同； 如果将单链表最后一个结点的指针指向头结点，使链表形成一个环形，此链表就称为循环链表（Circular Link List）。 （3）分组讨论，在实际应用中，单链表	1、观察图例演示和动画演示，了解循环链表和双向链表的结构特征，以及插入、删除等操作的内存变化过程； 2、讨论三种不同链表插入、删除和查找的步骤的不同； 3、根据图例，利用前导课程“C 语言	15min	**教学设计思路：** 1、仿真演示。通过对结点动态变化过程的演示，帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、分组讨论，帮助学生在讨论总结的过程中内化知识点，建立自己的知识架构； 3、思维拓展。 通过对不同链表存储结构的特点，在生活中寻找应用实例，培养学生在解决问题时进行多方面思考的能

	和单循环链表的使用环境 **2、双向链表** （1）虚拟仿真演示双向链表的结构形式：  （2）虚拟仿真演示双链表的插入运算：  q->prior=p; q->next=p->next; （p->next）->prior=q; p->next=q; （3）虚拟仿真演示双链表的删除运算  （p->prior）->next=p->next; （p->next）->prior=p->prior; free （p); （5）总结循环链表的结构特点； （6）对比单链表和单循环链表，分析三者之间的异同。包括不同操作需求下，三种链表的实际内存变化的区别； （7）分组讨论，在实际应用中，这三种链表的实际应用环境。	程序设计”中的“指针”知识点，用程序段描述地址变化过程； 4、根据三种不同链表的特点，讨论他们的应用环境。		力和习惯。面对具体问题能够灵活分析灵活处理。 **教学资源：** PPT 讲解，虚拟仿真，示例图演示。
课堂小结				
12	1、线性表是一种具有一对一的线性关系的特殊数据结构。 线性表有两种存储方法:用顺序存储方法来表示这种线性关系，得到顺序存储结构（即顺序表）;用链式存储方式来表示这种线性关系，得到线性表的链式存储结构（即链表); 2、线性表的链式存储结构，是通过结点之间的链接而得到的，链式存储结构有单链表、双向链表和循环链表等; 3、单链表结点至少有两个域:一个数据域和一个指针域。双向链表结点至少含有三个域:一个数据域和两个指针域; 4. 循环链表不存在空指针，最后一个结点的指针指向表头，形成一个首尾相接的	回顾本节课知识，构建知识架构。	2min	**教学设计思路：** 1、便于构成了一个完整的知识体系。使学生在自己的头脑中形成知识网络.从而达到提纲挈领的目的; 2、使学生带着问题进入教室，带着更多的问题离开教室，培养学生探究精神。

	环； 5. 为了处理问题方便，在链表中增加一个头结点； 6. 顺序存储可以提高存储单元的利用率，不便于插入和删除运算。链式存储会占用较多的存储空间，可以使用不连续的存储单元，插入、删除运算较方便。			
项目分析				
13	1、课堂连线企业导师，对上节课利用顺序表实现系统功能的项目实践进行点评； 2、对本节课利用链表实现系统功能进行项目分析，说明评价标准，强调注意事项。	根据企业导师点评进行自我反思，及时进行自我学习调整，分析本节课项目实现的设计要点，进行算法设计。	8min	**教学设计思路：** 企业导师根据行业标准进行项目分析及点评，帮助学生及时发现问题、解决问题，强化对知识的理解，提升学生的职业素养。

八、知识拓展

1、拓展训练

学生成绩统计表

学号	姓名	数学	物理	外语	总分
1	李华	88	89	90	267
2	王芳	98	90	87	275
3	张丽	78	84	90	252
4	田爽	89	69	78	236

用程序实现以下功能：
（1）插入新的一条记录“孟想，79，76，95，250”；
（2）删除第 3 条记录；
（3）查询王芳在第几条记录；
（4）依次输出表中所有记录。

2、考证（考研）训练：

（2012 年考研真题）关于线性表的顺序存储结构和链式存储结构，描述正确的是( )。
（1）线性表的顺序存储结构优于其链式存储结构。
（2）链式存储结构比顺序存储结构能更方便地表示各种逻辑结构。
（3）如频繁进行插入、删除结点操作，顺序存储结构更优于链式存储结构。
（4）顺序存储结构和链式存储结构都可以进行顺序存取。
A. （1）、（2）、（3） B. （2）、（4） C. （2）、（3） D. （3）、（4）

（软件水平考试）在单链表中,若 p 结点不是末尾结点，在其后插人 s 结点的操作是( )。

A. s-> next=p; p-> next=s; B. s-> next=p-> next; p-> next=s;

C. s-> next=p-> next; p=s; D. p-> next=s; s-> next=p;

3、竞赛训练

蓝桥杯线性表部分 https://dasai.lanqiao.cn/notices/1096/。

九、教学评价与反馈

本节课是数据结构第一种类型—线性结构的第二节课，涉及到链表的特征、在内存中的存储形式、以及在进行数据操作过程中内存地址的变化过程。理论知识较为抽象，需要学生加入个人的空间想象辅助理解。此外，在建立了空间逻辑结构的基础上还需要将其以程序的形式进行实现，因此，课程的实操性也很强。

由于在上一节课的学习中，学生已经掌握了线性结构和顺序表的基本操作，并实现了项目案例的所有功能。因此，本节课在讲授过程中，将“对比“贯穿始终。在讲单链表的时候，从链表的结构、存储方式到基本操作的程序实现，通过观看动画并引导学生讨论的方式，让大家主动的不断的与顺序表进行对比；在学习链表其他结构的时候，引导大家将当前数据结构与单链表相比，以头脑风暴的形式深化学生对链表不同结构的理解。而对于实践技能训练部分，通过小组讨论总结知识点、课堂回答问题、课堂练习等多种形式相结合的形式，每个学生都充分、全程并且深度参与课堂互动中，在体验、经验的基础上领悟、归纳、总结知识点，通过理解程序——仿写程序——知识迁移的形式逐步将抽象的数据结构形式转化为个人对程序的理解，并实现了其功能。达到了预想的教学效果。

由于学生已经利用线性表实现了项目功能，因此，在课后知识拓展部分，学生利用已掌握的顺序表的知识，对比新学习的链表知识，对系统功能进行分析，讨论系统实现的最佳数据结构，完善系统功能，对其进行优化。在此期间，既巩固了已学的知识点，又深化了对现学知识点的理解和把握，并通过对比，将不同数据结构与实践应用相结合，提升了学生的实际应用能力。

本节课存在的问题主要集中体现在双向链表的实际应用中。由于双向链表的操作包括两个指针域，学生在习惯了单链表操作的前提下，一时难以转换思路，利用逆向思维同时考虑两个指针的变换。

在后续的学习中，应该对前导课程“C 语言程序设计”指针和结构体两部分知识进行强化练习，提升基本程序的操作技能，此外，通过学习复合数据存储的范例程序，尝试解决类似的其他复合数据的相关功能，完成知识的拓展，深化理论知识理解。

十、教学总结

本节课采用项目教学法展开教学，课堂上通过一个简单的有序数据的增、删、查的基本操作，了解到在内存中数据的地址变换方式，将抽象的空间想象以具体的程序进行了实现，建立逻辑结构与功能实践之间的联系，深化理论知识理解，提升实践技能，并将“对比“这样的行为贯穿教学过程始终。

由于理论知识涉及到内存地址的变化，较为抽象，在讲解过程中采用“示例图讲解+动画演示+对比+学生自主总结”的方式进行理论知识讲解。学生通过与顺序表的对比，主动对链表进行总结，提升了学生的自我效能感，有效的将课堂知识内化到学生自身构建的知识体系中。

实践操作部分，由于学生在学习顺序表的时候已经完成了项目的所有功能的实现，对项目有了较为深刻的认知和理解。因此，本节课采用“对比”的方式，引导学生重新分析项目案例，从系统功能角度分析采用不同的数据结构，其算法复杂度和程序运行效率是否有所不同，从而重新对项目进行规划，优化系统功能。在这个过程中，一方面深化学生对知识的理解和灵活应用，另一方也锻炼了学生多角度全方位思考问题的习惯，提升了他们的实践技能和职业素养。

在整个课堂实施过程中，通过课堂回答问题、课堂练习以及小组讨论的形式，使学生全程参与课堂教学始终，保持了良好的、积极的课堂氛围。

十一、项目实现代码（关键代码段）

```
struct stu {
    int num ;
    char name[10];
    float s[3];
    char dj[3];
    struct stu *next ;
};

int input ()                          //   输入函数
{
     key_teacher ();//调用登陆界面函数

    struct stu *head=NULL,*pnew=NULL;
    char i;
    head=pnew=(struct stu *)malloc(sizeof (struct stu ));
    head->next =NULL;
    printf ("请输入学员成绩信息：\n");
    printf ("请输入学员的学号：\n");
    scanf ("%d",&pnew->num);
    printf ("请输入学员的姓名：\n");
    scanf ("%s",pnew->name );
    printf ("请输入学员的c语言成绩：\n");
    scanf ("%f",&pnew->s[0]);
    printf ("请输入学员的网络成绩：\n");
    scanf ("%f",&pnew->s[1]);
    printf ("请输入学员的英语成绩：\n");
    scanf ("%f",&pnew->s [2]);
    for (;;)
    {
        printf ("是否继续输入学员成绩信息( y / n )：\n");
        getchar( );
        i =getchar ( );
        if (i=='y'||i=='Y')
             pnew=creat(pnew);
        else if(i=='n'||i=='N')
        {
            save (head);
            break;
        }
        else
            printf ("您的输入有误，请重新输入，");
    }
    return 0;
```

```
}

struct stu *creat(struct stu *head)        //创建新节点
{
    struct stu *pnew=NULL;
    pnew=(struct stu *)malloc(sizeof (struct stu));
    head->next =pnew;
    pnew->next =NULL;
    printf ("请输入学员的学号：\n");
    scanf ("%d",&pnew->num);
    printf ("请输入学员的姓名：\n");
    scanf ("%s",pnew->name );
    printf ("请输入学员的c语言成绩：\n");
    scanf ("%f",&pnew->s[0]);
    printf ("请输入学员的网络成绩：\n");
    scanf ("%f",&pnew->s[1]);
    printf ("请输入学员的英语成绩：\n");
    scanf ("%f",&pnew->s [2]);
        return pnew;
}

int search ()              //    查找函数
{
    int a;
    struct stu *head=NULL;
    printf ("请输入您想要查询的学号：\n");
    scanf ("%d",&a);
    head=read();
    while (head!=NULL)
    {                                          //循环 找学号为a的同学
        if (head->num ==a)
        {
            printf ("该同学的成绩信息为：\n");
            djpaixu (head);        //调用等级函数，把成绩输出到屏幕上
            printf ("\n\n\n请按任意键继续...");
            getch ();
            system( "cls ");
            return 0;
        }
        head=head->next ;
    }
    printf ("对不起，没有学号为%d的同学的成绩信息！\n\n\n请按任意键继续...",a);
    getch ();
    system("cls");
    return 0;
```

```
}

int modify ()                                   //    修改函数
{   key_teacher ();                    //调用管理员界面
    int a,p=0;
    struct stu *head=NULL,*pnow=NULL,*head1=NULL;
    printf ("请输入您想修改信息的同学的学号：\n");
    scanf ("%d",&a);
    head1=head=read();
    while (head!=NULL)
    {
        if (head->num ==a)
        {   printf ("该同学的成绩为：\n");
            djpaixu (head);
            pnow=head;
            break;
        }
        head=head->next ;
    }
    if (pnow!=NULL)
    {   printf ("\n\n\n≮ 修改学号请输入.....1 ≯\n\n≮ 修改姓名请输入.....2 ≯\n\n≮ 修改C语言成绩请输入.....3 ≯\n\n≮ 修改网络成绩请输入.....4 ≯\n\n≮ 修改英语成绩请输入.....5 ≯\n\n≮ 返回主菜单请输入.....6 ≯\n\n");
        do
        {   scanf ("%d",&p);
            if(!(p>0&&p<7))
                printf ("您输入的数字有误，请重新输入：\n");
        }while (p<=0||p>=7);
        switch (p)
        {                    //写入新信息覆盖原有信息
        case 1:printf ("请输入新的学号：\n");scanf ("%d",&pnow->num );break;
        case 2:printf ("请输入新的姓名：\n");scanf ("%s",pnow->name );break;
        case 3:printf ("请输入新的C语言成绩：\n");
             scanf ("%f",&pnow->s[0] );break;
        case 4:printf ("请输入新的网络成绩：\n");
               scanf ("%f",&pnow->s[1] );break;
        case 5:printf ("请输入新的英语成绩：\n");
               scanf ("%f",&pnow->s[2] );break;
            case 6:break;
        }
        save1(head1);
        printf ("\n该学员成绩修改成功！\n\n请按任意键继续...");
        getch();
        system( "cls ");
        return 0;
```

```
    }
    else
        printf ("\n对不起，没有学号为%d的同学！\n\n\n\n请按任意键继续...",a);
    getch();
    system( "cls ");
    return 0;
}

int del()                          //              删除函数
{   key_teacher ();
    struct stu *head=NULL,*headold=NULL,*head1=NULL;
    int a=0;
    char i;
    printf ("请输入需要删除信息的同学的学号：\n");
    scanf ("%d",&a);
    headold=head1=head=read();
    while (head!=NULL)
    {
        if (a==head->num )
        {   printf ("该同学的成绩信息为：\n");
            djpaixu (head);
            break;
        }
        headold=head ;
        head =head->next ;
    }
    if (head==NULL)
    {   printf ("对不起，没有该同学的成绩信息！\n请按任意键继续...");
        getch();
        system( "cls ");
        return 0;
    }
    for (;;)
    {   printf ("\n是否要删除该同学的成绩信息( y / n )：");
        fflush(stdin);
        i=getchar();
        if (i=='y'||i=='Y')
        {
            if (head1==head)
            {head1=head->next ;        }
            else
            {headold->next =head->next ;          }
            save1(head1);
            printf ("该同学成绩信息删除成功！\n请按任意键继续...");
            getch();
```

```
            system( "cls ");
            return 0;
        }
        else if(i=='n'||i=='N')
        {
            system( "cls ");
            break;
        }
        else
            printf ("您的输入有误，请重新输入，");
    }
    return 0;
}
int djpaixu(struct stu *pnew)     //     按等级划分输出到屏幕上
{
    for (int y=0;y<3;y++)    //输出之前 判断三科成绩等级
    {
        if (pnew->s[y]>=90)                pnew->dj[y]='A';
        else if (pnew->s[y]>=80)           pnew->dj[y]='B';
        else if (pnew->s[y]>=70)           pnew->dj[y]='C';
        else if (pnew->s[y]>=60)           pnew->dj[y]='D';
        else
          pnew->dj[y]='F';
    }
    printf ("学号\t姓名\tC语言\t等级\t网络\t等级\t英语\t等级\n");
    printf ("%d\t%s\t%.2f\t %c\t%.2f\t %c\t%.2f\t %c\n\n",    pnew->num ,pnew->name ,
    pnew->s[0] ,pnew->dj[0],   pnew->s[1] ,pnew->dj[1],       pnew->s[2] , pnew->dj[2]);
    return 0;
}

int output ()          //查看函数：查看所有成绩
{
    int i;
    struct stu *head=NULL,*pnew=NULL;
    head=read();
    printf ("请选择排序输出的规则：\n");
    printf("\n≮ 按学号输出（由小到大）请输入.....1 ≯\n\n≮ 按C语言成绩输出（由大到
小）请输入.....2 ≯\n\n≮ 按网络成绩输出（由大到小）请输入.....3 ≯\n\n≮ 按英语成绩
输出（由大到小）请输入.....4 ≯\n\n");
    do
        {scanf ("%d",&i);
            if(!(i>0&&i<5))
            printf ("您输入的数字有误，请重新输入：\n");
        }while (i<=0||i>=5);
        switch (i)
```

```
        {
            case 1:
                    system( "cls ");
                    pnew=sort_xuehao(head);
                    while (pnew!=NULL)
                    {
                        djpaixu (pnew);
                        pnew=pnew->next ;
                    }
                    break;
            case 2:
                    system( "cls ");
                    pnew=sort_C(head);
                    while (pnew!=NULL)
                    {
                        djpaixu (pnew);
                        pnew=pnew->next ;
                    }
                    break;
            case 3:
                    system( "cls ");
                    pnew=sort_wangluo(head);
                    while (pnew!=NULL)
                    {
                        djpaixu (pnew);
                        pnew=pnew->next ;
                    }
                    break;
            case 4:
                    system( "cls ");
                    pnew=sort_english(head);
                    while (pnew!=NULL)
                    {
                        djpaixu (pnew);
                        pnew=pnew->next ;
                    }
                    break;
        }
        printf ("\n\n请按任意键继续...");
        getch();
        system( "cls ");
return 0;
}
```

《数据结构及算法设计》教案 3，7-8 学时

一、教学基本信息			
课程名称	栈	**授课教师**	余育丹
授课班级	21 级计算机应用工程 2 班	**授课时数**	2
授课时间	周三 1-2 节课	**授课地点**	81107

二、教学分析	
教学内容	1、理解栈的特点； 2、掌握栈的使用方法，能够对栈进行入栈、出栈； 3、能够根据栈的出栈顺序还原入栈顺序。
学情分析	本节课为第三章栈和队列的第一个知识点。在此之前，学生已经掌握了线性表的数据结构特点，并能较为灵活的使用顺序存储和链式存储分别实现项目的基本功能。栈作为线性结构的一种特殊应用，在具备线性表特点的同时，其独特的存取特点可以使得栈应用于一些特殊的功能实现。因此，本节课在讲解过程中，要注重将“对比”贯穿始终，从数据结构方面对比线性表的通用性和栈的特殊性，从存储方式方面对比顺序栈和链式栈的算法特点，从项目实现方面对比不同数据结构对于功能的优化。

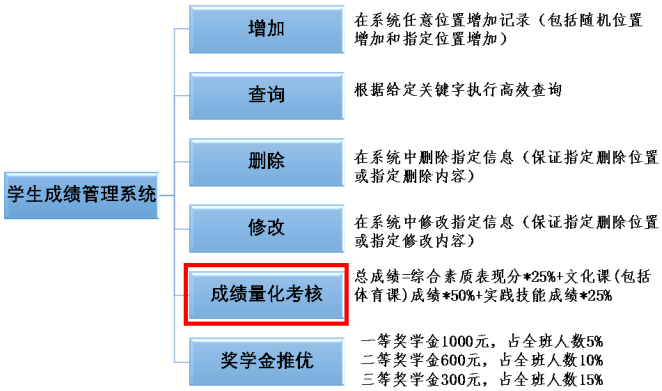
三、教学目标确定		
教学目标	**知识目标**	1、理解栈的定义和特点； 2、掌握栈底和顺序存储表示和链式存储表示； 3、掌握顺序栈空栈和满栈的判断条件、链栈和栈空条件。
	能力目标	1、熟练掌握栈的基本运算在顺序和链式两种存储结构下的运算实现； 2、利用栈编程的能力掌握数制转换、表达式求值等实例的解决方法； 3、具有独立学习，获取新知识和技能，在工作中发现问题、与分析问题、解决问题的能力。
	素质目标	1、提高对《数据结构》课程的学习兴趣，认识到栈在显示问题进行抽象中的重要作用； 2、培养理论联系实际、积极思考和自主训练的精神； 3、培养主动从生活中寻找科学原型的习惯。

教学重点	栈的特点，顺序栈和链栈上基本运算的实现算法，栈的简单应用。
教学难点	栈应用的实现算法，如数制转换、表达式求值。

四、思政融入课堂

通过学习先进后出这一工作原理，引导学生遵守社会秩序、尊重社会公德。

五、课程在项目中的定位

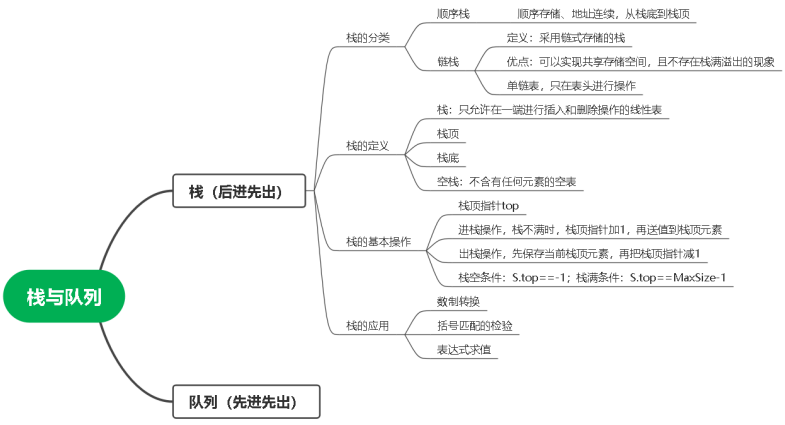


利用栈的表达式求值功能进行复杂的四则运算，实现“成绩量化考核”功能。

六、教学策略

设计思路

1、思维导图：



2、具体思路：

（1）问题引入：通过数制转换，迷宫求解，背包问题实例，引入新课；

（2）栈的定义：栈是一种特殊的线性表，是限定在表的一端进行插入和删除操作的线性表；

（3）栈的特点：栈的特点是“后进先出”；

（4）栈的出栈序列：通过实例，辅助虚拟仿真求解栈的出栈序列；
（5）栈的应用举例：通过实例，通过堆栈实现数制转换；
（6）拓展思维：用栈如何将中缀表达式转为后缀表达式；
（7）小结：内容总结。

3、教学方法：

采用 PBL 问题解决式教学法。为学生提供教材导读、学习任务、情景案例，以小组形式展开课前学习，以提出问题为目标，培养学生思维能力。教学中，实践项目提倡以小组协同合作的形式，通过头脑风暴、学生自主总结，教师从旁辅导的形式展开教学。教师根据不同小组的学情特征给予个性化的指导，完成授课环节。课后，引入小助教（学委、学习组长）机制。遵循当代大学生的心理特征，助教机制有效打破老师和学生之间的沟通障碍。

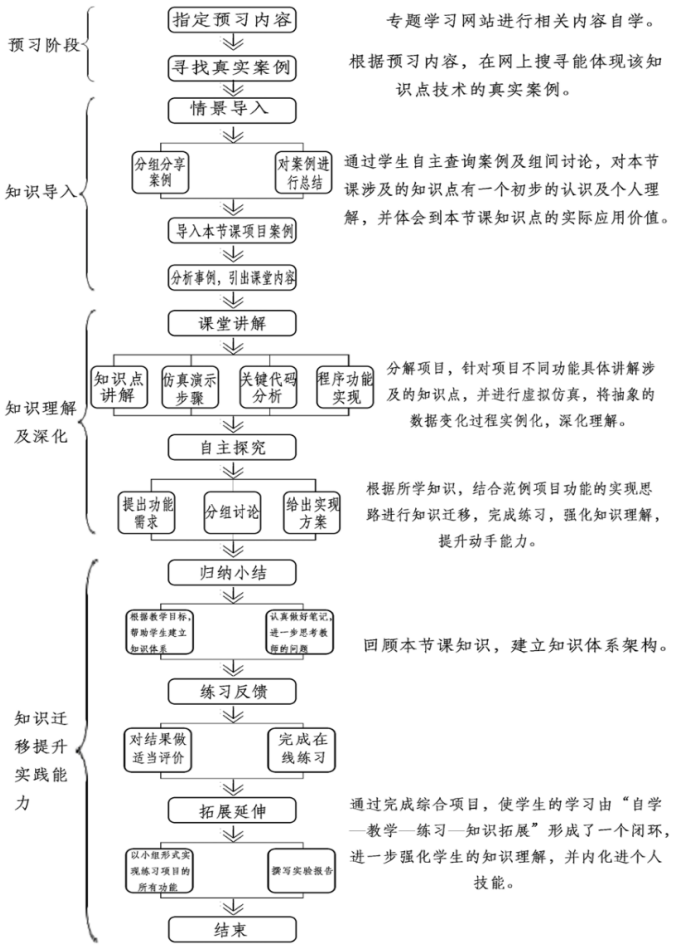
教学资源

学习环境：机房、局域网，交互式电子黑白

学习资源：

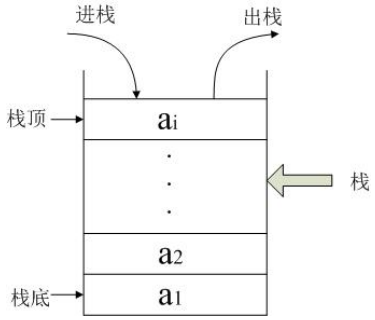
1、专题学习网站：包括慕课网、学习通等；
2、授课课件：根据此节学习内容制作的 PPT 课件；
3、多媒体资源库：虚拟仿真演示案例、项目演示案例；
4、案例库：课堂练习题库、测验题库等；
5、精品课程网站：本门课程的精品课程网站；
6、VC6.0++运行环境：进行案例演示及学生练习项目的专业运行环境。

教学流程



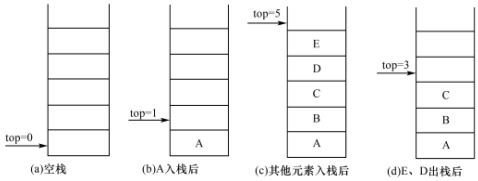
七、教学过程

教学环节	教学活动		时间安排	设计思路与教学手段
	教师授课内容	学生		
课前预习				
1	1、提供慕课网资源：请同学们预习本次课程基本知识，如栈的定义，特点等。 2、思政学习：未来计算机的革命 https://zhuanlan.zhihu.com/p/613298133 3、提出问题：栈的应用场景有哪些。	在充足的时间内观看慕课视频； 完成精品课成基础练习； 分组收集栈的应用案例。	课前一周	**设计思路：**利用慕课网等信息化手段进行自主学习，提高学生学习主动性和创造性，强化解决问题的能力。 **涉及教学资源：** 慕课网、搜索引擎。
知识导入				
2	**分析学生案例：** 探讨栈在现实生活中的应用。比如学生交本子：一叠本子也相当于一个栈，后交者先批，即后进栈者先出栈；栈在计算机系统也非常有用：常用栈来保存一些尚未处理和等待处理的数据项，对数据项的处理遵循“先进后出”的原则。	以组为单位分享栈的实际应用案例。	3min	**教学手段：** 分组讨论、真实案例。
3	**引出课堂案例：** 数值进位制的换算是计算机实现计算和处理的基本问题。将一个非负的十进制整数 N 转换为另一个等价的 h 进制数，很容易通过“除 h 取余法”来解决。其依据的基本原理是:N＝(N/h)×h＋N%h （其中，/为整除运算，%为求余运算)。以$(1998)_{10}=(3716)_8$为例，其转换过程如下: N N/8 N%8 1998 249 6 249 31 1 31 3 7 3 0 3	观察案例，思考如何用栈来解决。	2min	**设计思路：** 数制转换是学生在学习《计算机导论》课程中的内容，属于理论知识，选用学生熟悉的理论知识为案例，一方面学生熟悉其转换过程，不用过多的进行理论讲解，另一方面将理论知识用程序进行实现，激发学生的学习兴趣和探索欲望。

	请用程序实现数制转换过程。			
知识讲解				
4	**板书重要知识** **栈的定义和存储形式** **1、动画演示栈在内存中的地址特点，以及入栈和出栈的过程。** **2、栈的定义及结构特点：** 栈是一种运算受限制的线性表，其只允许在表的一端进行插入和删除操作，俗称堆栈。允许进行操作的一端称为“栈顶”，另一个固定端称为“栈底”，当栈中没有元素时称为“空栈”。 例如，栈(a 1 ,a 2 ,a 3 ,… ,a i)，其中 a 1 为栈底结点，而 a i 为栈顶结点。如果需要插入或删除结点，只能从栈顶操作，插入结点称为入栈，删除结点称为出栈，如图所示。  **3、讨论总结栈的行为特点：**后进先出。 **4、课堂练习：** 设将 a,b,c 三个字符依次进栈，最后都出栈，出栈可以在任何时刻（只要栈不空）进行，则出栈序列不可能是（ ）。	1、通过观看数制转换过程的仿真，了解栈的结构特点，以及执行入栈和出栈的行为过程； 2、思考栈的行为的主要分类，总结栈的特点； 3、依据线性结构及栈的学习经验，分析栈在进行具体数据存储中的各种可能性。	8min	**设计思路：**由于本节课涉及到栈的存取操作，其内部地址变化较为抽象，因此，该部分内容主要以虚拟仿真的形式进行讲解。用动画结合实例进行演示，帮助学生建立内存变化的直观印象。包括： 1、仿真演示。以顺序存储为例，演示栈在各种状态下的内存变化，帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、课堂讨论，头脑风暴。引导学生思考栈的行为，总结栈的变化规律和特点； 3、通过了解数制转换的内存变化过程，在抽象的空间想象与实际案例之间建立连接，完善知识架构； 4、课堂练习，巩固学生对栈的知识点的理解与应用。 **教学资源：** PPT 讲解，案例分析，虚拟仿真互动。
5	**以项目实现为目标,分解功能讲解顺序栈的运算** **1、顺序栈的定义：** 借助一维数组实现栈的存储。栈底的位置可以设置在数组的任何一端,而栈顶的位置是随着入栈和出栈操作而动态变化的。通常的做法是以数组下标为 0 的一端作为栈底，栈顶指针 top＝0 时表示空栈。	1、根据课堂讲解，结合第二章顺序存储结构的特	18min	**设计思路：**该部分教学内容涉及到较多程序代码的编写，以分组讨论、教师引导为主进行教学，主要包括： 1、对比与联系。由

2、课堂练习：

元素 A、B、C、D、E 入栈，再将 E、D 出栈后栈顶指针的变化情况。



3、顺序栈的类型说明：

```
#define Stack_Init_Size 100          //栈的初始大小
#define StackIncrement 10            //栈的存储空间分配增量
typedef int ElemType;                //ElemType为栈的元素类型
typedef struct
{
  ElemType*stackdata;   //栈中数据元素存储空间(一维数组)的起始地址
  int top;              //栈顶指针，实际上是栈顶位置的数组下标
  int stacksize;        //栈当前的可用大小，初始大小由Stack_Init_Size指定
}SeqStack;
```

4、栈的运算分类：

栈的运算指的是对栈中的数据进行操作，其具体实现与栈的物理结构有关。栈的基本运算有以下 9 种(除第一种，其余皆以栈 S 已存在为初始条件)：

(1) InitStack(&S)：构造一个空栈 S。

```
Status InitStack(SeqStack *S)
{
  if(!S)return Err_InvalidParam;              //顺序栈无效
  /*为stackdata分配内存空间*/
  S->stackdata=(ElemType*)malloc(Stack_Init_Size*sizeof(ElemType));
  if(!S->stackdata)returnErr_Memory;          //内存分配错误
  S->top=0;                                   //初始化栈顶指针为0
  S->stacksize=Stack_Init_Size;               //设置栈当前大小
  return OK;
}
```

(2) ClearStack(&S)：将栈 S 清为空栈。

```
Status ClearStack(SeqStack*S)
{
  if(!S)return Err_InvalidParam;              //顺序栈无效
  S->top=0;                                   //栈顶指针置为0
  return OK;
  }
```

(3) EmptyStack(&S)：测试栈 S 是否为空栈，若为空栈返回 True，否则返回 False。

```
int EmptyStack(SeqStack*S)
{
  if(!S)return Err_InvalidParam;              //顺序栈无效
  return(S->top==0);                          //top等于0为空，返回True，否则，
                                                不为空返回False
}
```

(4) LengthStack(&S)：求栈 S 的长度(即栈中数据元素的个数)。

```
int LengthStack(SeqStack*S)
{
  return S->top;                              //top指示栈中元素个数
}
```

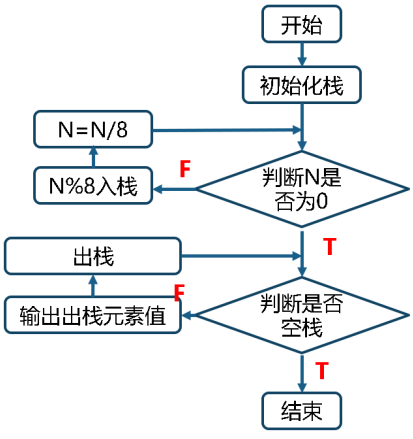
(5) TraverseStack(&S)：访问栈 S 中的每个元素一次。

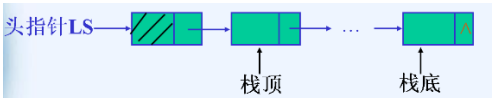
点，理解顺序栈的数据结构特点；
2、结合仿真动画，完成课堂练习，画出不同行为下栈的状态；
3、根据个人对栈的行为的理解，分析栈在 8 种运算中的状态特点，结合顺序栈的类型说明，小组讨论如何用程序对栈的状态进行描述，进而绘制程序流程图并编写程序代码。

于栈是线性表的一个特例，因此，结合线性表对栈进行讲解，更有助于学生理解，并逐步完善线性表的整体知识架构；
2、头脑风暴。学生通过栈变换过程的知识积累，讨论不同运算中栈的特点，并尝试编写代码段。这个过程可以帮助学生进一步内化抽象知识，在空间想象和实践之间建立联系；
3、思维拓展。教师在学生进行栈的 8 大算法设计分析的同时，引导学生思考极端情况的处理方法，思考其运行效率，培养学生在解决问题时进行多方面思考的能力和习惯。

教学资源：

PPT 讲解，示例图演示，案例分析。

	void TraverseStack(SeqStack*S) { int i; for(i=0;i<S->top;i++) printf("%d\t",S->stackdata[i]); } （6）GetTop(S,e)：获取栈 S 的栈顶元素，并将其存到 e 中。 Status GetTop(SeqStack*s,ElemType*e) { if(!S)return Err_InvalidParam; //顺序栈无效 if(S->top==0)return Err_NoResult; //栈为空 *e=S->stackdata[S->top-1]; //获取栈顶元素 return OK; } （7）Push(&S,e)：入栈操作，将元素 e 插入到栈 S 中，使其成为新的栈顶元素。 Status Push(SeqStack*S,ElemTypee) { ElemType*newstack; if(!S)return Err_InvalidParam; //顺序栈无效 if(S->top==S->stacksize) //栈满，追加存储空间 newstack=(ElemType*)realloc(S->stackdata, (S->stacksize+Stack_Increment)*sizeof(ElemType)); if(!newstack)return Err_Memory; //内存分配错误 S->stackdata=newstack; S->top=S->stackdata+S->stacksize; S->stacksize+=Stack_Increment; //修改当前栈的可用空间大小 S->stackdata[S->top++]=e; //将元素e存入栈顶，然后将top加1 return OK; } （8）Pop(&S,&e)：出栈操作，将栈 S 的栈顶元素删除，并将其存到 e 中。 Status Pop(SeqStack*s,ElemType*e) { if(!S)return Err_InvalidParam; //顺序栈无效 if(S->top==0)return Err_NoResult; //栈为空 *e=S->stackdata[--S->top]; //将top减1，然后将栈顶元素存入e中 return OK; }			
6	**项目实现：编写程序，实现**$(1998)_{10}=(3716)_8$的数制转换。 1、学生自主使用虚拟仿真软件查看数制转换的过程，讨论栈在其中的作用，口述转换过程。 2、知识迁移，根据步骤画出程序流程图：  3、根据程序流程图总结需要用到的栈的基本运算都有哪些。	1、观看数制转换的仿真过程，并分组讨论如何利用栈进行数制转换，总结转换步骤； 2、根据转换步骤绘制程序流程图； 3、小组讨论需要用到的基本算法；	15min	**教学设计思路：** 1、课堂讨论，口述数制的转换过程，帮助学生巩固顺序栈的特点，深化理论理解； 2、以任务为驱动，进行知识迁移与总结。程序的实现帮助学生在逻辑结构和功能实现之间建立连接，将理论知识转移为实践能力。此外，学生在实现程序过程中，将前面所学的关于栈的运算的零散知识点汇聚在一起，构建起了完整的关于顺

	4、选择合适的算法编写程序并实现。 5、分析程序算法复杂度。	4、课堂练习，实现数制的转换； 5、组间互评，分析算法复杂度。		序栈的知识架构，提高了编程的灵活性； 3、知识拓展，通过对程序进行算法复杂度分析，提出其优劣，培养学生全方位考虑问题的能力。 **教学资源：** PPT 讲解，示例图演示，编程环境。
7	**板书重要知识** **1、动画演示链栈在内存中的地址特点，以及入栈和出栈的过程。** **2、链式栈的定义：** 链式栈作为单链表的一种，其插入操作与删除操作均在链表头部进行，链表尾部就是栈底，头指针就是栈顶指针  **3、链栈结点结构的 C 语言描述：** typedef structnode { ElemType data; //数据域，用于存放栈中的数据元素 Struct node*next; //指针域 }StackNode,*LinkStack; **4、链栈的基本运算：** （1）获取栈顶元素 Status GetTop(LinkStack top,ElemType*e) { if(! top) returnErr_InvalidParam; //链栈无效 if(! top - > next) returnErr_NoResult; //栈为空 *e = top - > next - > data; //获取栈顶元素，并将其存入e中 return OK; } （2）元素入栈 Status Push(LinkStack top,ElemType e) { StackNode*s; if(! top)return Err_InvalidParam; //链栈无效 s = (StackNode*)malloc(sizeof(StackNode)); //生成新结点s if(! s)return Err_Memory; //内存分配错误 s - > data = e; //将数据元素e存放到新结点的数据域 s - > next = top - > next; //将s插入到栈顶 top - > next = s; return OK; } （3）元素出栈 Status Pop(LinkStack top,ElemType*e) { StackNode*p; if(! top)return Err_InvalidParam; //链栈无效 if(! top - > next)return Err_NoResult; //栈为空 *e = top - > next - > data; //将栈顶数据元素保存至e中 p = top - > next; //p指向栈顶结点 top - > next = p - > next; //删除栈顶结点 free(p); //释放结点 return OK; }	1、观察虚拟仿真动画，了解链栈的结构特点以及进栈和出栈的行为； 2、讨论链栈和顺序栈之间的不同，以及两者在实际使用中的优缺点； 3、根据链栈的数据结构特点，尝试用结构体进行描述； 4、结合链栈的行为特征，参考顺序栈的基本运算特征，用程序描述链栈的基本运算的程序段。	13min	**教学设计思路：** 链栈和顺序栈既相互区别又彼此联系。因此，在教授本节内容时，以“对比”为主，让学生在不断的将两者进行对比的过程中，一方面深化顺序栈的知识，另一方面进行知识迁移，引出链栈的运算规律。 1、虚拟仿真，仿真演示。演示链栈的结构特征以及基本的变化状态，帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、头脑风暴。结合链表和顺序栈的知识储备，对链栈的数据结构进行描述，并实现基本运算。在这个过程中，通过知识的迁移，帮助学生对链表有更深入的理解。 **教学资源：** PPT 讲解，案例分析，虚拟仿真互动。

8	**项目实现：编写程序，实现**$(1998)_{10}=(3716)_8$的数制转换。 1、根据之前的程序流程图分析需要用到的链栈的基本运算都有哪些； 2、选择合适的算法编写程序并实现； 3、分析程序算法复杂度，并于顺序栈的算法复杂度进行对比，分析两种不同栈结构的适用范围。	1、根据之前绘制的程序流程图讨论需要用到的基本算法； 2、课堂练习，实现数制的转换； 3、小组讨论，分析链栈和顺序栈的算法复杂度，并尝试总结两种不同栈结构的适用范围。	10min	**教学设计思路：** 1、知识迁移。由于之前已经对该程序有了充分的分析，因此，在本小节中，通过知识迁移的形式引导学生使用链栈进行程序实现，帮助学生形成一个完整的关于栈的知识架构； 2、对比联系 通过对链栈和顺序栈算法复杂度的分析和对比，可以帮助学生更全面的对栈有一个充分的认知，并在未来面对项目开发时，能够灵活的使用相应的数据结构进行功能实践，提升系统运行效率，提升个人实践技能。 **教学资源：** PPT 讲解，案例分析，编程环境。
9	**知识拓展：表达式求值** **1.算术表达式的中缀表示；** **2.算术表达式的后缀表示；** **3、虚拟仿真演示中缀表达式转换成等价后缀表达式算法。** 将中缀表达式转换成等价的后缀表达式后，表达式中操作数次序不变，运算符次序发生变化，同时去掉了圆括号。转换算法思路为:设立一个栈，存放运算符，首先栈为空，编译程序从左到右扫描中缀表达式: （1）若遇到操作数，直接输出，并输出一个空格作为两个操作数的分隔符； （2）若遇到运算符，则必须与栈顶比较，运算符优先级比栈顶级别高则入栈，否则栈顶元素出栈并输出； （3）若遇到左括号，入栈； （4）若遇到右括号，则一直出栈并输出，直到在栈中遇到左括号止。 **4、课堂练习：**	1、了解中缀表达式、后缀表达式的定义； 2、利用虚拟仿真技术，了解中缀表达式转后缀表达式的过程； 3、讨论总结栈在转换过程中的作用； 4、课堂练习，完	10min	**教学设计思路：** 选用贴近生活的案例，可以使学生充分感受到所学知识的价值，在实际项目开发过程中也能灵活采用相应的数据结构进行功能实现，增加学生的学习热情，丰富学生的开发经验。通过虚拟仿真动画演示复杂的转换过程，可以很大程度简化学生的理解，建立空间想象，也为学生自主绘制程序流程图，将知识内化提供条件。 **教学资源：** PPT 讲解，案例分

	将中缀表达式(12+2)*((38-2)/(7-4))转换成等价的后缀表达式。	整中缀表达式转后缀表达式； 5、小组总结转换的步骤，并尝试绘制程序流程图。		析，虚拟仿真。
课堂小结				
10	1、顺序栈的初始化就是将 top 设置为-1； 2、顺序栈判栈空操作就是判断 top 是否为-1； 3、顺序栈判栈满操作就是判断 top 是否为栈的最顶端位置； 4、链栈判栈空操作就是判断 top 是否为 NULL； 5、链栈的类型定义与单链表的类型定义相同； 6、链栈的初始化就是将 top 设置为 NULL。	回顾本节课的知识点，构建知识架构。	3min	**设计思路：**帮助学生构建一个完整的知识体系。解发学生更多更深的思考，完成拓展练习。
项目分析				
11	课堂连线企业导师，对上节课利用链表实现系统功能的项目实践进行点评；对本节课利用栈实现系统功能进行项目分析，说明评价标准，强调注意事项。	根据企业导师点评进行自我反思，及时进行自我学习调整，分析本节课项目实现的设计要点，进行算法设计。	8min	**教学设计思路：** 企业导师根据行业标准进行项目分析及点评，帮助学生及时发现问题、解决问题，强化对知识的理解，提升学生的职业素养。
八、知识拓展				
1、 拓展训练： 学生成绩管理系统“成绩量化考核”功能的实现。评定成绩的构成包括：				

（1）综合素质与表现分(占 25%):核算每个学生一学年内综合素质表现分”（两个学期平均分，小数点后保留一位)。一学年内“综合素质表现分”平均分超过 100 分的，按 25%比例折算时均计 25 分；
（2）各课（包括体育课）成绩（占 50%)：学年各课成绩以教学科研处所记载原始成绩为准，补考者按补考前成绩计，计算各课平均成绩（算术平均，小数点后保留两位)；
（3）实践技能成绩(占 25%) ：以教学科研处所记载实践技能原始成绩为准，补考者按补考前成绩计，计算实践技能平均成绩（算术平均，小数点后保留两位)；
将以上三项成绩代入以下公式计算总成绩(小数点后保留两位)：
总成绩=综合素质表现分 x25%+文化课(包括体育课)成绩 x50%+实践技能成绩 x25%

2、考证（考研）训练：

（2019 年考研真题）下列关于栈的叙述中，错误的是（ ）。
I.采用非递归方式重写递归程序时必须使用栈
II.函数调用时，系统要用栈保存必要的信息
III.只要确定了入栈的次序，即可确定出栈次序
IV.栈是一种受限的线性表，允许在其两端进行操作
A.仅 I B.仅 I、II、III C.仅 I、III、IV D.仅 II、III、IV

（软件水平考证）若元素 a,b,c,d,e,f 依次进栈，允许进栈、退栈操作交替进行，但不允许三次进行退栈操作，则不可能得到的出栈序列是（ ）
A. dcebfa B. cbdaef C. bcaefd D. afedcb

3、竞赛训练：

蓝桥杯栈部分 https://dasai.lanqiao.cn/notices/1096/

九、教学评价与反馈

本次栈内容的教学涵盖了栈的定义、常见运算、类型等多个方面，从概念、原理、实现等多个角度深入浅出地介绍了栈的相关知识点。整个教学过程中，采用了举例、动画演示等多种教学方式，让学生能够更好地理解和掌握栈的相关知识。

学生在学习本节课之前，已经运用顺序表和链表分别实现了学生成绩管理系统的大部分基本功能，对线性结构有了较为充分的了解，所以在学习本节课时，通过观看仿真动画，结合线性结构的知识积累，能够较为容易的进行知识迁移，了解并掌握栈的运算规律，总结算法特点。因此，学生在基本程序练习方面完成的较好，针对数制转换问题也能够用程序进行实现。

表达式求值是栈的另一项重要应用。也是“学生成绩管理系统”中，对成绩量化考核功能进行优化的重要手段。要求学生利用所学栈的知识，对成绩量化考核功能设计更为复杂、公平的计算方法并实现。这部分内容作为拓展内容，要求学生在课后完成。从学生及企业反馈信息来看，大部分学生能够利用栈的思想优化该功能，但仍有部分学生无法实现这样较为综合的栈的应用。可见，虽然学生能够编写出栈的不同状态的代码段，但理解不够透彻，没有办法将零散的知识点灵活运用于不同的算法需求中，其整体知识架构的搭建有待完善。

针对本章的教学，我认为可以做出以下改进和优化：

1. 更加注重实践：在讲解栈的实现原理和相关算法时，可以加强对实际场景的应用，让学生更加深入地理解栈的实际使用场景和应用价值。

2. 优化教学方式：在讲解某些概念时，可以采用更加形象、生动的教学方式，如图表、实物等，帮助学生更好地理解和记忆相关知识点。

十、教学总结

栈是处理实际问题及开发各种软件时经常使用的数据结构，运算受限的特殊的线性表，但从数据类型角度看，和线性表大不相同的抽象数据类型。栈是限定只能在表的一端（栈顶）进入插入与删除的线性表，其特点是先进后出。

本节课除了讨论栈的定义、表示方法和实现外，还给出一些应用的例子。从同学们的学习及作业情况来看，本节课的理解比上一章要好很多，经分析理由如下，一是本章内容相对上一章简单，二是经过上一章的学习后，栈的应用就显得简单，是否可以考虑将第三章的学时压缩2个学时，增加到第二章。

提高课后上机操作的任务量，充分吸收本章课内容，为后续队列学习打下良好的基础。

十一、项目实现代码

```
#include <stdio.h>
#include <malloc.h>
#include <string.h>
#include "Stack.c"                                          //栈实现程序
int ComparePriority(char Lopt, char Ropt)
{
  int LPriority=0,RPriority=0;                              //分别保存 Lopt 和 Ropt 的优先数
  /* 设置 Lopt(左边运算符)的优先数 */
  if(Lopt=='+'||Lopt=='-') LPriority=1;                     //加和减的优先数为 1
  else if(Lopt=='*'||Lopt=='/') LPriority=2;                //乘和除的优先数为 2
  /* 设置 Ropt(右边运算符)的优先数 */
  if(Ropt=='+'||Ropt=='-') RPriority=1;                     //加和减的优先数为 1
  else if(Ropt=='*'||Ropt=='/') RPriority=2;                //乘和除的优先数为 2
  /* Lopt 的优先数大于等于 Ropt 的优先数返回 True,否则返回 False */
  return ((LPriority-RPriority)>=0);
}
/* InfixToSuffix(Infix)函数将中缀表达式 Infix 转换成相应的后缀表达式 */
char* InfixToSuffix(char * Infix)
{
  char ch1,ch2, chpre='\0';                 //chpre 用于存放 Infix 当前字符的前一个字符
  char * Suffix;                            //存放转换的后缀表达式
  int posIn=0,posSu=0;                      //分别指示 Infix 和 Suffix 当前位置
  SeqStack * Soptr;                         //声明栈
  Suffix=(char*)malloc((strlen(Infix)+1) * 2* sizeof(char));   //分配空间
  if(! Suffix) {printf("内存分配失败!\n");return NULL;}
  Soptr=(SeqStack*) malloc(sizeof(SeqStack));               //为栈 Soptr 分配地址
  if(! Soptr) {printf("内存分配失败!\n") ;return NULL;}
  InitStack(Soptr);                                         //初始化栈
  while(Infix[posIn]! ='\0')                                //逐一扫描字符串 Infix
    {
      ch1=Infix[posIn];                     //ch1 为获取的 Infix 的当前字符
      if(ch1==' '){posIn++;continue;}       //如果当前字符为空格,则忽略
      if(ch1>='0'&&ch1<='9')                //如果当前字符为数字
      {/* 如果前一个字符是运算符或左括号,Suffix 当前位置填入一个空格,以表示新操作数的开
始,同时将当前的数字字符填入下一位置 */
        if(chpre=='+'||chpre=='-'||chpre=='*'||chpre=='/'||chpre=='(')
        {
          Suffix[posSu++]=' ';              //在新操作数前填入空格
          Suffix[posSu++]=ch1;
                                            //数字字符直接进入 Suffix 的当前位置,posSu 加 1
        }
        else if(chpre=='\0'||(chpre>='0'&&chpre<='9'))
        {
          Suffix[posSu++]=ch1;              //数字字符直接进入 Suffix 的当前位置,posSu 加 1
        }
```

```
    {printf("表达式非法！\n");return NULL;}
  }
  else if(ch1==='(')
    Push(Soptr,ch1);                          //如果是左括号,直接入栈
  else if(ch1===')')                  //如果是右括号,遇到左括号前将栈中运算符出栈添加到 Suffix
  { /*栈不空,且栈顶元素非左括号*/
    while(! EmptyStack(Soptr)&&(GetTop(Soptr,&ch2) ==OK) &&ch2! = '(')
    {
     Suffix[posSu++]=ch2;                  //栈顶运算符填入 Suffix
     Pop(Soptr, &ch2);                     //出栈
    }
   if(EmptyStack(Soptr))
   { printf("表达式非法！\n");return NULL;}//没有遇到左括号
     Pop(Soptr, &ch2);                     //左括号出栈
   }
   else if(ch1==='+'||ch1==='-'||ch1==='*'||ch1==='/')     //如果当前为运算符
   {
     if(EmptyStack(Soptr)) Push(Soptr,ch1);//栈空则直接入栈
     else
     {/*否则将栈中所有优先级高于当前运算符的运算符依次出栈,将其添加到 Suffix 中*/
       while(! EmptyStack(Soptr)&&(GetTop(Soptr,&ch2)==OK)&&ComparePriority(ch2, ch1))
       {
         Suffix[posSu++]=ch2;              //栈顶运算符填入 Suffix
         Pop(Soptr, &ch2) ;                //运算符出栈
       }
       Push(Soptr,ch1) ;                   //将当前运算符入栈
     }
   }
     else { printf("表达式非法！\n");return NULL;}    //若 ch 为其他字符,表达式非法
     chpre=ch1;                            //chpre 保存 Infix 的当前字符
     posIn++;                              //posIn 指向 Infix 的下一字符
   }
   while(! EmptyStack(Soptr))              //将栈中其余的运算符依次出栈,并添加到 Suffix 中
   {
     Pop(Soptr, &ch2);
     Suffix[posSu++ ]=ch2;
   }
   Suffix[posSu]='\0';                 //为 Suffix 设置字符串结束标识
   free(Soptr);                        //释放 Soptr
   return Suffix;                      //返回后缀表达式 Suffix
}
int StrToInt(char *str)
{
   int i,result=str[0]-48;
   for(i=1;i<strlen(str);i++)
     result=result *10+(str[i]-48);
   return result;
}
/*计算两个操作数的结果*/
int Cal(int opnd1, char opt, int opnd2)
{
   int result;
   switch(opt)
   {
     case '+':result=opnd1+opnd2;break;          //加法运算
     case '-':result=opnd1-opnd2;break;          //减法运算
     case '*':result=opnd1*opnd2;break;          //乘法运算
     case '/':result=opnd1/opnd2;break;          //除法运算
     }
     return result;
}
```

```
/*以下函数计算输入的后缀表达式结果,成功返回计算结果,失败则返回-1*/
int CalculateExpression(char * Suffix)
{
  char ch, stropnd[11];          //stropnd 用于临时存放要转换成整数的操作数字符串
  int opnd1,opnd2,i,pos=0;
  SeqStack *Sopnd;                                    //声明栈
  Sopnd=(SeqStack *)malloc(sizeof(SeqStack));         //为栈 Sopnd 分配地址
  if(! Sopnd){printf("内存分配失败! \n");return -1;}
  InitStack(Sopnd);                                   //初始化栈
  while(Suffix[pos]! ='\0')                           //扫描后缀表达式 Suffix
  {
    ch=Suffix[pos];                                   //ch 为后缀表达式当前字符
    if(ch==' '){pos++ ;continue;}                     //空格略过
    if(ch>='0'&&ch<='9')                              //如果当前字符是数字
     {/*将连续的数字组合成一个整数(表达式中操作数之间是由空格或其他运算符隔开的)*/
     i=0;
     while(Suffix[pos]>='0'&&Suffix[pos]<='9')
       stropnd[i++]=Suffix[pos++];                    //将连续的数字存储到字符串 stropnd
     stropnd[i]='\0';                                 //设置字符串结束符
     opnd1=StrToInt(stropnd);                         //将字符串 stropnd 转换成整数
     Push(Sopnd, opnd1);                              //操作数入栈
     }
   else                                               //如果不是数字,则是运算符
   {/*从栈中弹出两个操作数*/
     if(Pop(Sopnd, &opnd2) ! =OK)
     { printf("表达式非法! \n"); return -1;}
     if(Pop(Sopnd, &opnd1) ! =OK)
     { printf("表达式非法! \n"); return-1;}

      Push(Sopnd, Cal(opnd1.ch, opnd2));              //将两个操作数计算结果入栈
      pos++ ;  //pos 指向下一字符
    }
  }
if(Pop(Sopnd, &opnd1) ! =OK||! EmptyStack(Sopnd)) //将结果出栈,保存到 opnd1
  { printf("表达式非法! \n"); return -1; }        //如果出栈失败或出栈后栈不空,表达式非法
    free(Sopnd) ;                                   //释放栈
    return opnd1;                                   //返回结果
}
/*以下为主程序*/
void main()
{
  char Expression[100], *Suffix;
  int result;
  printf("请输入计算表达式:");
  gets(Expression) ;                                  //输入表达式字符串(中缀)
  Suffix=InfixToSuffix(Expression) ;                  //将中缀表达式转换成后缀表达式
  if(Suffix)
   {
    printf("后缀表达式为:%s\n", Suffix);
    result=CalculateExpression(Suffix);               //调用后缀表达式求值算法计算结果
    if(result>=0)                                     //计算成功
      printf("运算结果为:%d\n", result);
  }
}
```

《数据结构及算法设计》教案 4，9-10 学时

一、教学基本信息			
课程名称	队列	**授课教师**	佘育丹
授课班级	21 级计算机应用工程 2 班	**授课时数**	2
授课时间	周五 3-4 节课	**授课地点**	81107

二、教学分析	
教学内容	1、队列的顺序存储表示和链式存储表示； 2、顺序存储结构下循环队列的表示和实现； 3、链队的表示和实现； 4、队列的应用。
学情分析	本节课为第三章栈和队列的第二个知识点。在此之前，学生已经掌握了线性表的数据结构特点，并能较为灵活的使用顺序存储和链式存储分别实现项目的基本功能。队列作为线性结构的一种特殊应用，在具备线性表特点的同时，其独特的存取特点可以使得队列应用于一些特殊的功能实现。因此，本节课在讲解过程中，要注重将“对比”贯穿始终，从数据结构方面对比线性表的通用性和队列的特殊性，从存储方式方面对比队列的顺序存储和链式存储的算法特点，从项目实现方面对比不同数据结构对于功能的优化。

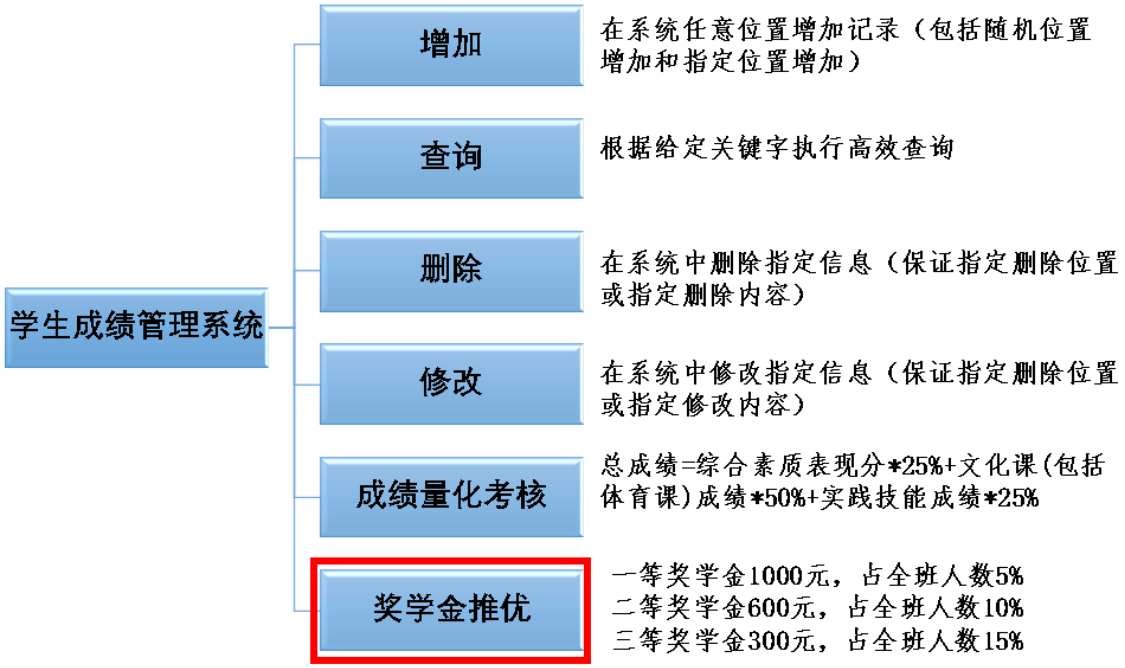
三、教学目标确定		
教学目标	**知识目标**	1、理解队列的定义和特点，包括先进先出（FIFO）的原则； 2、理解队列的实现方式，包括数组和链表两种常见的实现方式； 3、掌握队列的基本操作，包括入队、出队、判空和获取队首元素等操作。
	能力目标	1、能够设计和实现基本的队列数据结构，包括利用数组或链表实现队列的存储和基本操作； 2、能够分析和解决基于队列的算法问题，例如广度优先搜索、队列排序等； 3、能够对队列的实现方式和操作进行优化，例如使用循环队列等方法提高队列的效率。
	素质目标	1、培养学生具备辩证思维的能力； 2、培养学生具有热爱科学、勇于实践、实事求是的学风和创新意识、创

		新精神； 3、培养学生具有较强的执行能力以及较高的工作效率和安全意识； 4、培养学生规范、严谨、精确的工作态度和情感。
教学重点		理解队列的定义、属性和基本操作，并能够设计和实现基本的队列数据结构。
教学难点		理解队列的实现方式和操作的效率，以及分析和解决基于队列的算法问题。

四、思政融入课堂

队列作为一种常用的数据结构，其工作的基本原理是 FIFO，即先进先出，后进后出。知识讲解过程中对这一工作原理的理解比较简单，但在教学过程当中，会把遵守社会秩序、尊重社会公德的内容引申给学生，引导学生树立起正确的社会导向。

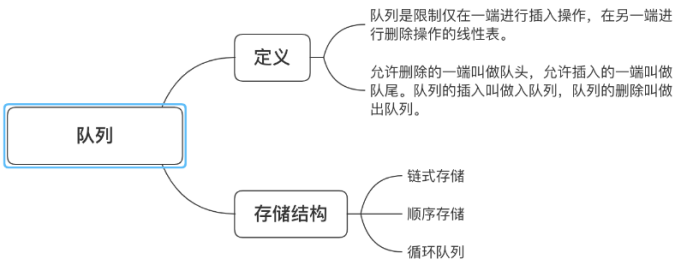
五、课程在项目中的定位



利用队列“先进先出”的功能，根据学生的总分情况，从学生成绩表中分别筛选出一等奖、二等奖和三等奖的奖学金获得者。

六、教学策略

设计思路	1、思维导图：



2、具体思路：

（1）问题引入：通过 Josephsus 问题引入新课；

（2）队列的定义；

（3）队列的特点：先进先出；

（4）队列的顺序实现：一般情况，极端情况，解决方案（借助虚拟仿真演示）；

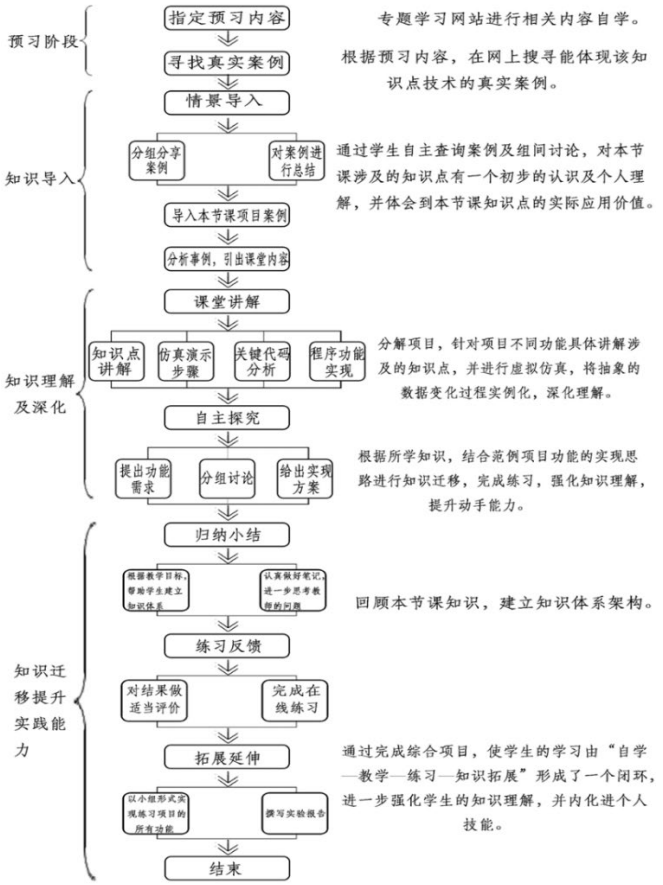
（5）队列的链式实现，完成项目实践；

（6）拓展思维：队列的其他应用；

（7）内容小结，对比线性表、栈、队列结构的特点，加深学生对队列的理解。

3、教学方法：采用 PBL 问题解决式教学法。为学生提供教材导读、学习任务、情景案例，以小组形式展开课前学习，以提出问题为目标，培养学生思维能力。教学中，实践项目提倡以小组协同合作的形式，通过头脑风暴、学生自主总结，教师从旁辅导的形式展开教学。教师根据不同小组的学情特征给予个性化的指导，完成授课环节。课后，引入小助教（学委、学习组长）机制。遵循当代大学生的心理特征，助教机制有效打破老师和学生之间的沟通障碍。

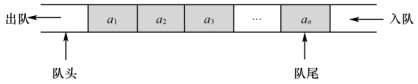
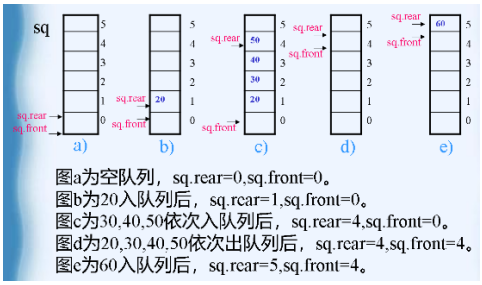
教学流程

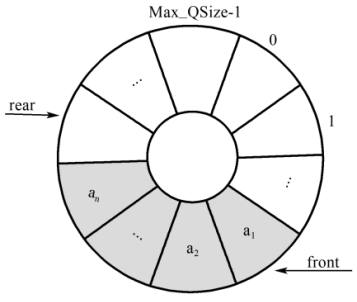


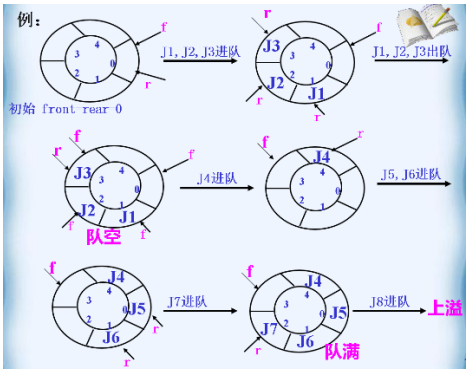
教学资源	**学习环境**：机房、局域网，交互式电子黑白 **学习资源：** 1、专题学习网站：包括慕课网、学习通、职教云等； 2、授课课件：根据此节学习内容制作的 PPT 课件； 3、多媒体资源库：虚拟仿真演示案例、项目演示案例； 4、案例库：课堂练习题库、测验题库等； 5、精品课程网站：本门课程的精品课程网站； 7、VC6.0++运行环境：进行案例演示及学生练习项目的专业运行环境。

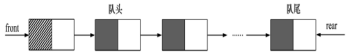
七、教学过程

教学环节	教学活动		时间安排	设计思路与教学手段
	教师授课内容	学生		
课前预习				
1	1、提供慕课网资源，请同学们预习本次课程基本知识，如队列的定义，特点等； 2、思政学习：区块链技术 https://zhuanlan.zhihu.com/p/434508958 3、提出问题：队列的应用场景有哪些？	1、在充足的时间内观看慕课视频； 2、完成精品课成基础练习； 3 分组收集队列的应用案例。	课前一周	**设计思路：**利用慕课网自主学习，提高学生学习主动性和创造性，强化解决问题的能力。 **涉及教学资源：** 慕课网、搜索引擎。
知识导入				
2	1、回顾上节课内容：栈的顺序存储结构和链式存储结构； 2、分析学生案例，探讨队列在现实生活中的应用。	以组为单位分享队列的实际应用案例。	3min	**教学手段：** 分组讨论、真实案例。
3	**引出课堂案例：** 假设 n 个人围坐在一圈，并按顺时针方向 1~n 编号。现在从某个人开始进行 1~m 报数，数到第 m 个人时，此人出圈;接着从他的下一个人重新开始 1～m 报数，数到第 m 个人时，此人也出圈;如此进行下去，直到所有人都出圈为止。试设计算法给出这些人的出列	观察案例，思考如何用队列来解决。	2min	**设计思路：**选用世界经典案例激发学生的学习兴趣和积极性。

	顺序表。 由于该问题是由古罗马著名史学家 Josephus 提出的问题演变而来的，所以称为 Josephus 问题。			
	知识讲解			
4	**板书重要知识点：** **1、动画演示队列在内存中的地址特点以及入队和出队的过程。** **2、队列的定义** 队列是一种基本的数据结构，是一种具有先进先出（FIFO）特性的线性数据结构。它的定义可以简单概括为：队列是一种只允许在一端插入数据，在另一端删除数据的线性表。 例如，我们可以将队列类比为一条排队等候的人群，新来的人只能站到队尾，而最先到的人只能先走出队列。这样，我们就可以通过队列这种数据结构来实现类似排队、调度等各种应用场景。 **3、队列的术语** 如图所示，对于具有 n 个元素的队列 $Q=(a_1,a_2,\cdots,a_n)$，称 a1 为队头元素，a_n 为队尾元素。队列中的元素按 a1，a2，---，a_n 的顺序进队，也只能按 a_1，a_2，---，a_n 的顺序出队，即队列的操作是按照先进先出原则进行的。  **4、讨论总结队列的行为特点：**先进先出 **5、课堂练习** 设将 a,b,c 三个字符依次入队，最后都出队，总共有几种出队形式？	1、通过观看仿真动画，了解队列的结构特点，以及执行入队和出队的行为过程； 2、思考队列行为的主要分类，总结队列的特点； 3、依据线性结构及队列的学习经验，分析队列在进行具体数据存储中的各种可能性。	8min	**设计思路：** 1、仿真演示。以顺序存储为例，演示队列在各种状态下的内存变化，帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、课堂讨论，头脑风暴。引导学生思考队列的行为，总结队列的变化规律和特点； 3、课堂练习，巩固学生对栈的知识点的理解与应用； 4、思政教育。引导同学们，在现实生活中，要尊守社会秩序，先到先服务，不要扰乱公共秩序，遵守社会公德。 **教学资源：** PPT 讲解，案例分析，虚拟仿真互动。
5	**顺序队列的定义及特点** **1、学生根据掌握的队列知识，口述入队及出队过程：**  ； **2、顺序队列的结构体定义**	1、根据课堂讲解，结合第二章顺序存储结构的特点，理解顺序队列的数据结构特点； 2、观察队列的行为特征，尝试用	8min	**设计思路：** 1、课堂讨论，头脑风暴。通过总结队列的行为特点，帮助学生巩固理论知识。利用程序对其进行描述，帮助学生在抽象的空间想象与实际案例之间建立连接，完善知识架构； 2、极端问题。极端

	`` #define Max_ QSize 100 //队列最大容量(分配空间大小) typedef struct {ElemType data[Max_QSize]; //队列数据元素存储空间 int front; //头指针(实际上是 data 数组的下标) int rear; //尾指针(实际上是 data 数组的下标) } SeqQueue; //顺序队列类型 ``` ； **3、根据队列的变换过程，尝试用程序描写入队、出队的行为，以及判断队空和队满的条件。** 入队列操作： sq.rear=sq.rear+1; sq.data[sq.rear]+x; 出队列操作： sq.front=sq.front+1; 队空条件： q->front==q->rear 队满条件： q->rear== maxsize -1; 4、反思本例中的情况 e，思考能否继续入队； 5、通过对极端情况的分析，了解“假溢出”状态，思考解决方案。	程序代码对队列的行为进行描述； 3、思考极端条件，组内讨论解决方案。		问题的提出可以引导学生逐步将程序设计与实际情况相结合，培养学生全面思考问题的习惯。同时，问题的产生也会引起学生强烈的求知欲，保持课堂的活跃度 。 **教学资源：** PPT 讲解，案例分析。
6	**循环队列的定义和特点** **1、动画演示循环队列在内存中的地址特点，以及入对和出对的过程；** **2、循环队列定义及结构特点** 循环队列将队列的数据区假想成是首尾相连的环(如图所示)，即将存储队列元素的一维数组的最小下标和最大下标位置假想成物理相邻的两个存储空间，允许队列的头、尾指针从最大下标位置直接前进到最小下标位置。在循环队列中，无论是出队还是入队操作，头指针或尾指针都要增 1，只不过如果指针增加 1 后达到了队列的最大分配空间大小时，应将该指针设为 0，这就是所谓的“循环意义下加 1”。指针 p 循环意义下加 1 可以用 p 加 1 后取队列最大长度的模来实现，即 p＝(p＋1)％Max_QSize。  在 C 语言中通过一维数组来实现循环队列，使用时用户必须为队列设置一个最大容量(长度)，如果无法预知队列使用的最大长度，则最好采用链队列。下面来讨论循环队列的一	1、观看仿真动画，总结循环队列的行为特点； 2、与普通队列进行对比，通过观察循环队列的行为特征，用程序对其关键行为进行描述。	8min	**设计思路：** 1、课堂讨论，头脑风暴。通过总结循环队列的行为特点，帮助学生巩固理论知识。利用程序对其进行描述，帮助学生在抽象的空间想象与实际案例之间建立连接，完善知识架构； 2、对比式教学。不断对比循环队列与队列，可以帮助学生从已有的知识中进行知识迁移，完成新知识的学习，并自主构建其知识架构。 **教学资源：** PPT 讲解，案例分析，虚拟仿真互动。

	些主要基本操作的实现算法。 **3、图例演示循环队列行为特点**  **4、知识延审，**总结队列和循环队列特点； **5、知识迁移，**描述循环队列入队、出队的行为，以及判断上溢和下溢的条件： 入队列操作： Sq.rear=(sq.rear+1)%maxsize 出队列操作： Sq.front=(sq.front+1)%maxsize 上溢条件： (CQ.rear+1)%maxsize==CQ.front 下溢条件： CQ.front==CQ.rear。			
7	**知识拓展：** 编写算法模拟舞伴配对问题，先入队的男士或女士亦先出队配成舞伴。 **1、项目分析** 首先设置两个队列(Mdancers, Fdancers)，分别存放男士和女士入队者。假设男士和女士的记录存放在一维数组中作为输入，然后依次扫描该数组的各元素，并根据性别来决定是进入男队还是女队。当这两个队列构造完成之后，依次将两队当前的队头元素出队来配成舞伴，直至某队列变空为止。此时，若某队仍有等待配对者，算法输出此队列中排在队头的等待者的名字，他(或她)将是下一轮舞曲开始时第一个可获得舞伴的人。 **2、绘制程序流程图。** **3、项目实践。**	1、了解程序功能需求分析； 2、组内讨论程序流程图，确定算法； 3、上机实践，并进行算法分析。	15min	**设计思路：**培养学生的团队合作精神和团队意识； 帮助学生将所学知识内化为程序代码，提高实践动手能力。 **教学资源：** PPT 讲解，案例分析。
8	**链队** **1、动画演示链队在内存中的地址特点，以及入队和出队的过程；** **2、链队的定义**	1、观察虚拟仿真动画，了解链队的结构特	15min	**设计思路：** 链队和顺序队既相互区别又彼此联系。因此，在教授本节内容

	用链表表示和存储的队列称为链队列。链队列的结点结构与单链表相同，只不过队列的插入和删除操作分别是在队尾和队头进行的，因此除了设置指示队头的头指针(front)外，还要设置指示队尾的尾指针(rear)。为了方便起见，同线性表一样，我们采用带有头结点的链表来表示队列。队头是由头结点的指针域指示的，而且当头指针和尾指针都指向头结点时，表示队列是空队列。  **3、链队的结构体定义** `typedef int ElemType; //数据元素类型为整型` `/*结点结构*/` `typedef struct node` `{` `ElemType data; //数据域` `struct node * next; //指针域` `}QueueNode;` `/*链队列*/` `typedef struct` `{` `QueueNode *front; //头指针` `QueueNode *rear; //尾指针` `}LinkQueue ; //链队列类型` **4、链队的基本运算** （1）初始化队列； （2）判断队列是否为空； （3）获取队头元素； （4）元素入队； （5）元素出队； （6）销毁队列。	点以及进队和出队的行为； 2、讨论链队和顺序队之间的不同，以及两者在实际使用中的优缺点； 3、根据链队的数据结构特点，尝试用结构体进行描述； 4、结合链队的行为特征，参考顺序队的基本运算特征，用程序描述链队列的基本运算的程序段。		时，以“对比”为主，让学生在不断的将两者进行对比的过程中，一方面深化顺序队的知识，另一方面进行知识迁移，引出链队的运算规律： 1、虚拟仿真，仿真演示。演示链队的结构特征以及基本的变化状态，帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、头脑风暴。结合链表和顺序队的知识储备，对链队的数据结构进行描述，并实现基本运算。在这个过程中，通过知识的迁移，帮助学生对链表有更深入的理解； 3、思政教育。由于队列与日常生活中的排队情况本质一致，可引申到遵守社会秩序、尊重社会公德的层面，进而对学生进行社会社义核心价值观教育。 **教学资源：** PPT 讲解，案例分析，虚拟仿真互动。
9	**项目实践 Josephus 问题** 1、根据需求说明，讨论链队在其中的作用，口述解决方案； 2、知识迁移，根据步骤画出程序流程图； 3、根据程序流程图总结需要用到的链队的基本运算都有哪些； 4、选择合适的算法编写程序并实现； 5、分析程序算法复杂度。	1、分组讨论如何利用队列解决 Josephus 问题，总结步骤； 2、根据步骤绘制程序流程图； 3、小组讨论需要用到	20min	**教学设计思路：** 1、课堂讨论，口述解决方案，帮助学生巩固链队的特点，深化理论理解； 2、以任务为驱动，进行知识迁移与总结。程序的实现帮助学生在逻辑结构和功能实现之间建立连接，将理论知识转移

		的基本算法； 4、课堂练习，实现 Josephus 问题； 5、组间互评，分析算法复杂度。		为实践能力。此外，学生在实现程序过程中，将前面所学的关于队列的运算的零散知识点汇聚在一起，构建起了完整的关于队列的知识架构，提高了编程的灵活性； 3、知识拓展，通过对程序进行算法复杂度分析，提出其优劣，培养学生全方位考虑问题的能力。 **教学资源：** PPT 讲解， 示例图演示，编程环境。
课堂小结				
10	1、理解队列的定义和特点，掌握队列的顺序存储表示(循环队列)和链式存储表示。由于队列容易产生“假溢出”(“假队满”)现象，因此常采用循环队列作为队列的顺序存储结构； 2、在算法设计方面，要求熟练掌握栈的基本运算在顺序和链式两种存储结构下的运算实现;熟练掌握循环队列的基本运算的实现，循环队列判断队空和队满的条件，以及带头结点链队的入队、出队、置空队等操作的实现;并能够根据栈和队列的基本运算，自己设计一些简单的应用程序并上机调试和运行。	回顾本节课的知识点，构建知识架构。	3min	**设计思路：**帮助学生构建一个完整的知识体系。解发学生更多更深的思考，完成拓展练习。
项目分析				
11	课堂连线企业导师，对上节课利用栈实现系统功能的项目实践进行点评；对本节课利用队列实现系统功能进行项目分析，说明评价标准，强调注意事项。	根据企业导师点评进行自我反思，及时进行自我学习调整，分析本节课项目实现的设计要点，进行算法设计。	8min	**教学设计思路：** 企业导师根据行业标准进行项目分析及点评，帮助学生及时发现问题、解决问题，强化对知识的理解，提升学生的职业素养。

八、知识拓展

1、拓展训练：

在学生成绩管理系统中，实现“统计输出奖学金获得者名单”功能，要求保证先进先出的原则，总成绩高的学生优先评定奖学金。

总成绩=综合素质表现分×25%+文化课（包括体育课）成绩×50%+实践技能成绩×25%。

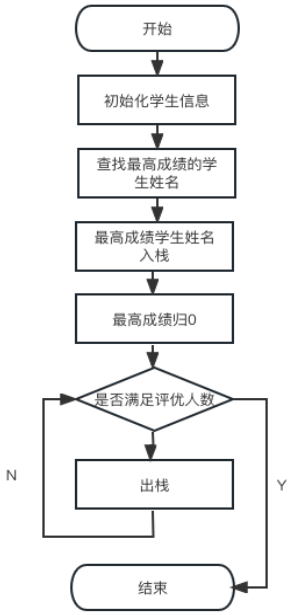
奖学金奖励金额与比例按如下规则：

一等奖学金 1000 元，占全班人数 5%

二等奖学金 600 元，占全班人数 10%

三等奖学金 300 元，占全班人数 15%

使用队列，实现奖学金推优功能。



2、考证（考研）训练：

（2019 年考研真题）已知头指针 h 指向一个带头结点的非空单循环链表，结点结构为 data next，其中 next 是指向直接后继结点的指针，P 是尾指针，q 是临时指针。现要删除该链表的第一个元素，正确的语句序列是(　　)。

A　h -> next = h -> next -> next; q = h-> next; free(q);

B　q = h -> next; h -> next = h -> next -> next; free(q);

C　q = h-> next; h -> next = q -> next; if(p != q)p = h; free(q);

D　q = h-> next; h -> next = q-> next; if(p == q)p = h; free(q);

（软件水平考证）

若以 abcd 作为双端队列的输入序列，则既不能由输入受限的双端队列得到，也不能由受限的双端队列得到的输出序列是（　　）。

A　abcd　　B　dacb　　C　dbca　　D　dbac

3、竞赛训练：

蓝桥杯队列部分 https://dasai.lanqiao.cn/notices/1096/。

九、教学评价与反馈

本次队列内容的教学涵盖了队列的定义、常见运算、类型等多个方面，从概念、原理、实现等多个角度深入浅出地介绍了队列的相关知识点。整个教学过程中，采用了举例、动画演示等多种教学方式，让学生能够更好地理解和掌握队列的相关知识。

针对本章的教学，我认为可以做出以下改进和优化：

1、更加注重实践：在讲解队列的实现原理和相关算法时，可以加强对实际场景的应用，让学生更加深入地理解队列的实际使用场景和应用价值；

2、优化教学方式：在讲解某些概念时，可以采用更加形象、生动的教学方式，如图表、实物等，帮助学生更好地理解和记忆相关知识点；

3、增加练习：在讲解完队列的基本概念和运算后，可以适当增加一些练习题，让学生能够更好地巩固所学知识点，并在实践中掌握队列的实际应用。

总的来说，本章的教学内容全面、系统，教学方法多样，但也存在一些可以进一步优化的地方，希望能够不断改进和完善，让学生更好地掌握队列的相关知识。

十、教学总结

在本章的学习中，我们介绍了数据结构中的队列。队列是一种基本的数据结构，具有先进先出的特点，广泛应用于计算机科学和软件工程中。

在本章的学习中，我们掌握了队列的定义、属性和基本操作，包括入队、出队、判空和获取队首元素等操作。我们还介绍了队列的两种常见的实现方式：数组和链表，并讨论了它们各自的优缺点。

在本章的学习中，我们还学习了一些基于队列的算法和应用，例如广度优先搜索、队列排序等。我们了解了如何分析和解决基于队列的算法问题，并学会了如何优化队列的实现方式和操作。

总之，本章的学习让我们深入了解了队列这一基本数据结构的定义、属性和基本操作，并让我们掌握了设计和实现基本队列数据结构、分析和解决基于队列的算法问题以及优化队列实现方式和操作等实用技能。

十一、项目实现代码

Josephus 问题的完整程序如下：

```
# define NULL 0                          /* 循环链接队列的简单应用实例——Josephus 问题 */

# define LEN sizeof(linkqueue)
# include "stdio.h"
# include "BHCLEAR.c"                    /* BHCLEAR.c 文件包含 clear 清屏、定位等函数 */
typedef struct node                      /* 结点的类型定义 */
{ int data;                              /* 标识号数 */
  struct node *next ;                    /* 指针 */
} linkqueue;                             /* 循环链接队列的类型说明 */
linkqueue *creat_ rcir(),*report_ num();      /* 函数说明 */
int people, num;                         /* people 为循环队列总人数,num 为出列报数 */
```

```
linkqueue *creat_rcir()                    /*循环链接队列建立模块——尾插法建表 */
{ int i;                                   /*建立单循环链接队列,并给每个人编号*/
  linkqueue *head, *p, *rear;
  head=(struct node*)malloe(LEN);          /*建立第一个结点*/
  head->data=1;                            /*第一个结点的编号为 1 */
  rear=head;                               /*尾指针初值为 head */
  for(i=2;i<=people; i++)                  /*输入 people 个数据*/
  { p=(struct node*)malloc(LEN);           /*生成新结点*/
    p->data=i;                             /*输入新结点的编号*/
    rear->next=p;                          /*新结点插到表尾 rear */
    rear=p;                                /* rear 指向新的表尾*/
  }
  rear->next=head;                         /*将表尾结点指向头结点*/
  return (head);                           /*返回循环链表头指针*/
}/* CREATE_CIRCLE */

linkqueue *report_num(head, people)        /*报数出列模块——循环链接队列删除函数*/
linkqueue *head;                           /*每个人报数出列,直到队列为空为止*/
int m,people;                              /* m 为报数出列数, people 为总人数*/
{ linkqueue *p1, *p2;
  int max, j, count;
  max=people;
  printf("\n\n\t\t 报数出列的顺序为:");
  count=1; j=0;                            /* count 统计报数,j 统计出列人数*/
  p1=head; p2=p1;                          /* p2 是指向 p1 前驱结点的指针*/
  do{ p2=p1->next;                         /* p2 指向当前报数的人*/
    count=count+1;                         /*报一次数*/
    if(count%m==0)                         /*满足报数条件,则此人出列*/
    { j++;                                 /*一人出列*/
      printf("\n\t j= %2d\tno= %2d",j,p2->data);  /*打印出列号数*/
      p1->next=p2->next;                   /*报数出列,从队列中删除该结点*/
      free(p2);                            /*释放结点存储空间*/
      }


      else p1=p2;                          /* p1 后移一个结点继续报数*/
}while(j! =max);                           /*报数出列,直到所有人出列为止*/
head=NULL;                                 /*将队头指针设置为空指针*/
printf("\n\n\t 报数完毕再见!");             /*程序结束*/
return (head);                             /*函数返回链表头指针*/
}/* REPORT_NUM */

main()                                     /*循环链队列简单应用实例——Josepues 问题主程序*/
{ linkqueue *hcir, *head;
  clear();                                 /*清屏幕函数*/
  printf("\n\n\t\t 请输入出列报数 num=");    /* 输入出列报数 num */
  scanf("%d", &num) ;                      /* num 为出列报数*/
  printf("\n\t\t 请输入循环队列总人数 people="); /*输入队列总人数*/
  scanf("%d", &people);                    /* people 为队列人数*/
  hcir=creat_rcir() ;                      /*建立循环链接队列*/
  head=report_ num(hcir, num, people) ;    /*报数出列*/
}/* MAIN */
```

《数据结构及算法设计》教案 5，11-12 学时

一、教学基本信息			
课程名称	树和二叉树	**授课教师**	申妙芳
授课班级	21 级计算机应用工程 3 班	**授课时数**	2
授课时间	周一 7-8 节	**授课地点**	81107

二、教学分析	
教学内容	树是具有层次或嵌套关系的非线性结构，被广泛用于计算机领域，尤其二叉树最重要、最常用。本章是数据结构课程的重点之一，是后续许多章节的基础，本节课的内容包括树的基本概念，二叉树的定义、性质和存储表示及相关算法的实现。这些内容理论性强，知识点多，内容又高度抽象，需要适当放慢点速度，多引入实际生活例子类比，帮助同学们理解吸收。
学情分析	本节课是树的第一章节内容，是另一种数据结构。由于其一对多的结构特点，学生在建立空间想象方面会有一定的困难，尤其在将树的逻辑结构与物理存储结构转换方面，很多学生都难以理解，从而直接影响数据结构课程的教学效果，打击学生的学习热情。因此，在讲解理论知识的时候，尽量选用贴近学生生活的案例进行对比讲解，帮助学生理解专业术语；在讲解树的存储行为时，以项目案例为驱动，结合虚拟仿真、图例演示、头脑风暴、组内讨论等教学手段，帮助学生建立空间想象，并与实际项目相结合，将逻辑思维落地到具体的程序实现上，在这个过程中深化学生的理论知识，提升他们的实践技能。

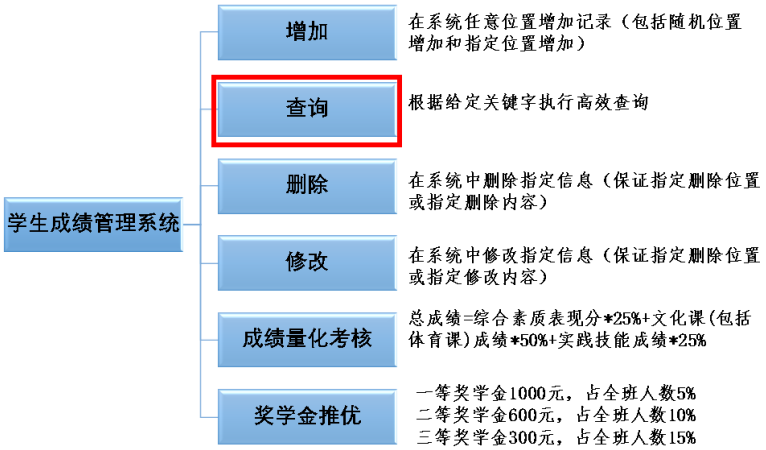
三、教学目标确定		
教学目标	**知识目标**	1、理解树的定义和特点； 2、掌握二叉树的概念和性质； 3、掌握二叉树的存储结构和基本操作。
	能力目标	1、熟练掌握树、二叉树的定义和相关术语，熟练掌握二叉树的性质和相应的证明方法； 2、掌握二叉树的顺序存储结构和链式存储结构； 3、提高学生信息素养和意识，引导学生正确应用所学算法理论解决实际问题的能力。
	素质目标	1、树立科学发展观，养成实事求是的科学态度，具有观察、理解、判断、推理的辩证思维能力；

		2、培养学生对抽象概念的理解能力； 3、培养学生发现问题、思考问题和解决问题的能力。
教学重点		树型结构的概念，二叉树的定义、存储结构及建立。
教学难点		二叉树的定义、存储结构及建立，二叉树的非递归运算及应用。

四、思政融入课堂

根据树结构特点引申出家族、家谱的概念，通过讲解家谱的发展历程，鼓励学生学习传统文件，辩证地传承传统文化，取其精华，去其糟粕。

五、课程在项目中的定位

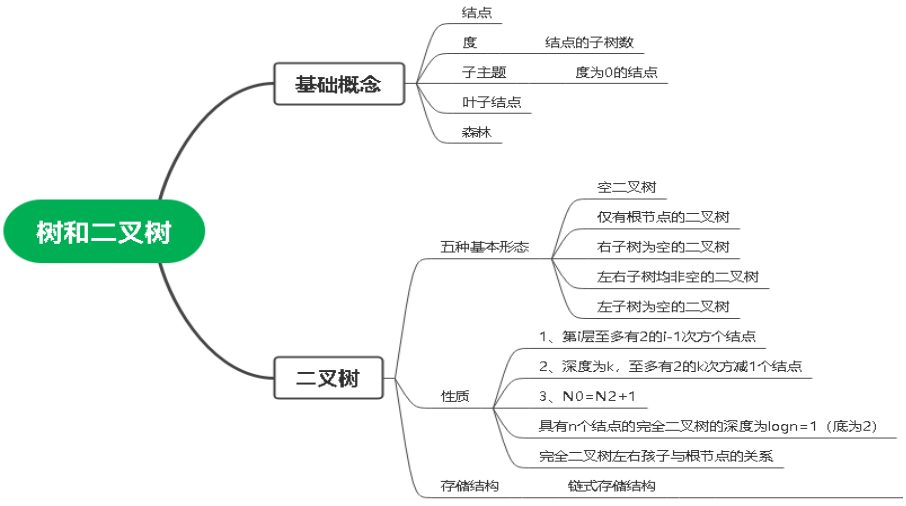


拓展“查询”功能，利用树形结构按照考核等级进行分类，以层级式输入学生相关信息。

六、教学策略

设计思路

1、思维导图：



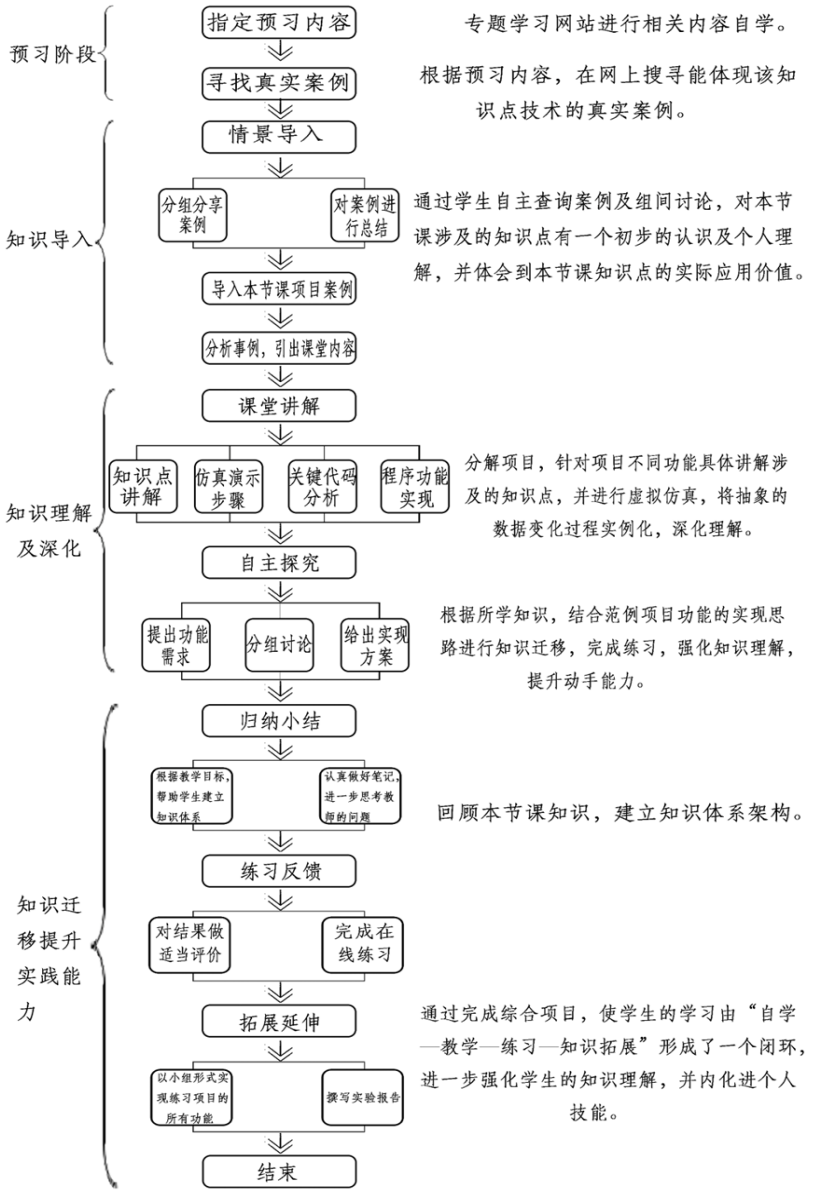
2、具体思路：

课前预习树的基本知识，让大家了解概念，识记基础简单知识点。课中以案例的形式引入新课，例如数据压缩问题，利用二叉树求解表达式值的问题。课堂中提出问题，同学们进行讨论。根据树结构特点引申出家族、家谱的概念，通过讲解家谱的发展历程，鼓励学生学习传统文件，辩证地传承传统文化，取其精华，去其糟粕。

3、教学方法：

本节主要采用启发式教学。课前发布一些应用管理类系统的视频，让同学们直观感受到树的应用价值，激发同学们求知欲望。在教学中，有意识地培养学生的创新思维能力，可以提高学生理论联系实际的能力、发现问题以及灵活独特地解决问题的能力。在讲解树的概念时可以类比家族关系，树的相关术语在家族关系中的体现，查找家族中的某个成员过程就是树的查找算法，再比如组织结构图等，先列举现实生活中的例子，这些例子都与这些概念有着密切的关系，这样学生就很容易接受并记住这些概念。

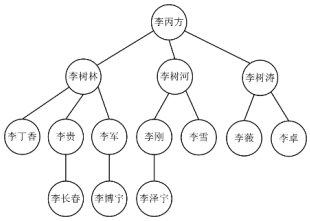
教学流程

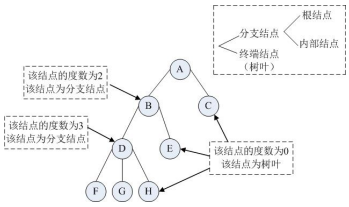
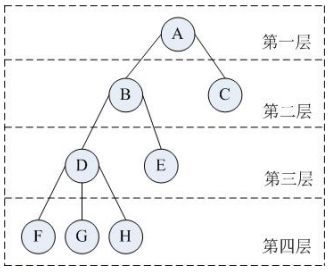


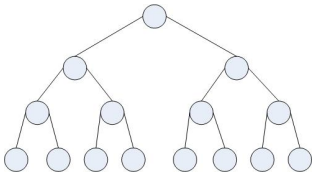
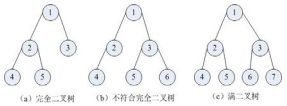
教学资源	**学习环境：**机房、局域网，交互式电子黑白 **学习资源：** 1、专题学习网站：包括慕课网、学习通、职教云等； 2、授课课件：根据此节学习内容制作的 PPT 课件； 3、多媒体资源库：虚拟仿真演示案例、项目演示案例； 4、案例库：课堂练习题库、测验题库等； 5、精品课程网站：本门课程的精品课程网站； 6、VC6.0++运行环境：进行案例演示及学生练习项目的专业运行环境。

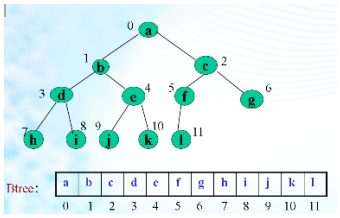
七、教学过程

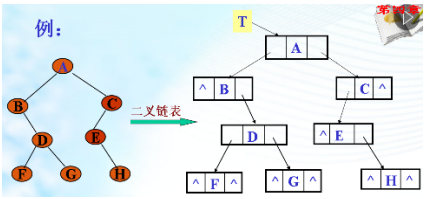
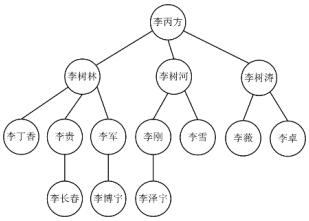
教学环节	教学活动 教师授课内容	学生	时间安排	设计思路与教学手段
课前预习				
1	1、提供慕课网资源，请同学们预习本次课程基本知识，如树的定义，基本操作，二叉树的性质和基本操作等； 2、提出问题：二叉树的应用场景有哪些？	1、在充足的时间内观看慕课视频； 2、完成精品课成基础练习； 3、分组收集二叉树的应用案例。	课前一周	**设计思路：**利用慕课网自主学习，提高学生学习主动性和创造性，强化解决问题的能力。 **涉及教学资源：** 慕课网、搜索引擎。
知识导入				
2	**回顾上节课内容：**队列的基本运算，注意事项。 **分析学生案例，**探讨树在现实生活中的应用。	以组为单位分享树的实际应用案例。	3min	**教学手段：** 分组讨论、真实案例。
3	**引出课堂案例：** 在日常生活当中，家族中经常都有家谱，如图所示，该家谱共有四代人，其中李丙方属于最高辈分，李长春、李博宇和李泽宇属于最低辈分，那么我们如何存储家谱中的人员信息和人员之间的关系呢？例如：若输入李泽宇，则输出父亲是李刚，若输入李树林，则输出孩子是李丁香、李贵和李军。	观察案例，思考如何存储家谱中的人员信息，人员之间的关系又怎样反映。	2min	**设计思路：**激发学生的学习兴趣和积极性。

				
4	**分析案例，引出新课：** 家族家谱属于树形结构，我们可以采用双亲表示法存储家谱数据，首先确定家族家谱中的总人数，根据人数定义结构体类型，每个结点应该包含姓名和双亲两个域，创建家谱就是输入姓名和双亲编号。若输出某人的父亲，首先要输入某人姓名，找到此人后记录下双亲编号，再通过双亲编号输出父亲姓名；若输出某人的孩子，首先要输入某人姓名，找到此人后记录下此人对应的下标，再输出双亲等于该下标的孩子姓名。	观察案例，思考家谱在计算机中怎样存储。	2min	**设计思路：**引导学生学习，发挥学生主观能动性。
知识讲解				
5	**板书重要知识：** **树的定义：** 1、引导学生回忆数据结构导论部分对树的结构的介绍，与线性结构对比，并总结树的结构特点 树（Tree）是 N（N≥0）个结点的有限集合，它满足以下 4 个条件。 （1）有且只有一个特定的被称为根（Root）的结点； （2）除了根结点，其余每个结点都有且只有一个直接前驱； （3）每个结点都可以有多个后继，没有后继的结点称为叶结点（树叶）； （4）除了根结点，其他结点可以分为 m（m≥0）个互不相交的有限集合 T1、T2……Tm，其中每一个集合又可以视为一棵树，称为根的子树（SubTree）。 **树的基本术语** **1、知识迁移** 结合自然界的树和家族亲属关系，利用学生的个人社会经验对树的术语进行理解、记忆。 （1）度数	1、根据已有知识讨论总结树型结构的特点 2、将个人对自然界树的认知和家族亲属关系的认知迁移至树的术语学习中，帮助记忆和理解	10min	**设计思路：** 1、对比教学： 树型结构中很多专业属于来自于自然界的树。因此，在进行专业术语讲解时，将树与自然界的树进行对比，利用学生的生活经验完成知识迁移，构建关于树的知识架构。 培养学生的主观能动性和创新能力。 **教学资源：** PPT 讲解，案例分析，

	一个结点的子树（或后继结点）的个数称为该结点的度数。度数为 0 的结点称为叶结点或终端结点，度数不为 0 的结点称为分支结点，除根结点以外的分支结点称为内部结点。一棵树的度数指的是该树中结点的最大度数。如图所示。  （2）结点关系 结点的子树的根（结点的后继结点）称为该结点的孩子（Child），同时该结点称为孩子结点的双亲结点（父结点），具有同一双亲结点的孩子结点互相称为兄弟结点。 图中，结点 B 的孩子结点为结点 D 和结点 E，换句话说，结点 B 是结点 D 与结点 E 的双亲结点，结点 B 与结点 C 互为兄弟结点。 （3）结点层次 树也可以被视为一种层次结构，树中的每个结点都在固定的层次上。结点层次从根结点开始定义，根结点的层次为 1，其孩子结点的层次为 2，以次类推。树中结点的最大层次称为树的深度（Depth）或高度。如图所示。  （4）有序树与无序树 兄弟结点有顺序（不可交换）的树称为有序树，兄弟结点无顺序的树称为无序树。			
6	**思政教育** 由树结构的特点引出思政教育。	积极参与思政话题讨论。	2min	**设计思路：** 教育学生爱家、爱国。 **教学资源：** 案例分析。
7	**1、二叉树的概念** 二叉树是一种特殊的树形结构，其中的每一个结点最多只能有两个直接后继。二叉树的递归定义如下。 （1）有且只有一个根结点；	1、回忆树的结构特点，进行知识迁移，根据二叉树的概念讨论二叉树	8min	**设计思路：** 1、知识迁移。引导学生利用树的知识进行迁移，理解并总结树的特殊形态——二叉

	（2）可以是空树，当为非空树时，它由一个根结点以及两棵互不相交且分别称为左子树和右子树的二叉树组成。 二叉树的任意结点最多只有两棵子树，也可以没有子树或者只有一棵子树，因此二叉树的度数一定小于或等于 2。二叉树严格区分左右子树，即使只有一棵子树也要区分左右。 综上所述，二叉树具有以下 5 种基本形态。 （1）空二叉树； （2）只有一个根结点； （3）一个根结点和根结点的左子树构成； （4）一个根结点和根结点的右子树构成； （5）一个根结点和根结点的左右子树构成。 **2、满二叉树** 满二叉树每层的结点数都是最大结点数，除了最后一层为叶结点，其余所有结点都有左右子树。深度为 k 的满二叉树有 2 k -1 个结点，如图所示。  **3、完全二叉树** 对一棵具有 n 个结点的二叉树的结点按层序进行编号，如果编号为 i（1≤i≤n）的结点与同样深度的满二叉树的中编号为 i（1≤i≤n）的结点位置相同，那么该树就是完全二叉树。  **4、 二叉树的性质** （1）性质 1 二叉树的第 i（i≥1）层中最多有 2 i-1 个结点。图所示的满二叉树中，第 3 层结点的个数为 2 3-1=4 个，第 4 层结点的个数为 2*4-1=8 个。 （2）性质 2 深度为 k 的二叉树最多有 2 k-1 个结点。在同等深度的二叉树中，满二叉树的结点数最多，叶结点数最多。	的结构特点 2、组内总结二叉树和树之间的区别与联系 3、讨论二叉树的几种极端形态，总结满二叉树和完全二叉树之间的区别与联系 4、逻辑推导二叉树的性质		树的特点，内化知识，完善知识架构。 2、对比教学 通过对比行为，可以引导学生自主总结不同形态树之间的区别与联系，深化知识理解，完善知识架构。 培养学生的主观能动性和创新能力。 **教学资源：** PPT 讲解，案例分析，图例演示。

	（3）性质 3 在任何一棵二叉树中，如果叶结点的数量为 N，度数为 2 的结点数量为 N 2，则 N=N 2+1。假设该树中度数为 1 的结点数量为 N 1 ，则这棵树的总结点数为 N+N 1 +N 2 ，总结点数也可以是所有子结点数加 1（根结点）的和，即 N 1 +2×N 2 +1。因此 N 1 +2×N 2 +1=N+N 1 +N 2 ，得出 N=N 2 +1。			
8	1、课堂练习，职教云平台； 2、课堂讨论：一棵完全二叉树有 5000 个结点，可以计算出其叶结点的个数是（ ）。	1、互助学习； 2、小组讨论。	10min	**设计思路：** 现学现练，有助于巩固学生的理论知识，深化理解 培养学生的团队合作精神和团队意识。 **教学资源：** PPT 讲解，职教云平台，题库。
9	**二叉树的存储** 1、二叉树的顺序存储 （1）观看虚拟仿真动画，了解二叉树顺序存储的特点 使用顺序存储实现二叉树就是用一维数组存储二叉树中所有的结点，并通过数组的下标体现结点在二叉树中的位置。  （2）提出极端情况，引导学生思考顺序存储的适用范围，总结优缺点 2、二叉树的链式存储 （1）观看虚拟仿真动画，了解二叉树链式存储的特点 在顺序存储不适用的情况下，可以考虑使用链式存储。使用链式存储表示二叉树，其结点结构与双向循环链表一致，即一个数据域和两个指针域。	1、观看虚拟仿真动画，了解树的顺序存储和链式存储的行为特点； 2、对两种存储方式进行对比，总结适用范围； 3、用程序段分别描述两种存储的行为。	17min	**设计思路：** 1、虚拟仿真。本节内容是课程的重点也是难点。由于涉及到树形结构转换为线性结构在内存中进行存储，内容较为抽象，难以通过语言建立空间想象。利用虚拟仿真动画，帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、对比与练习。通过对两种存储形式的对比，可以帮助学生深化对知识的理解，构建树的知识架构； 3、知识迁移。利用线性表中顺序存储和链式存储的相关知识，可以帮助学生更好的理解树的运算，并用程序进行描述。 **教学资源：** PPT 讲解，

	 （2）思考链式存储的结点定义； （3）根据链式存储的行为特征，讨论并总结其基本运算的实现。			示例图演示，编程环境。
10	**项目实现，代码演示。**  1、根据需求说明，讨论二叉树在其中的作用，口述解决方案； 2、知识迁移，根据步骤画出程序流程图； 3、根据程序流程图总结需要用到的树的基本运算都有哪些； 4、选择合适的算法编写程序并实现； 5、分析程序算法复杂度。	1、分组讨论如何利用二叉树解决族谱输入问题，总结步骤 2、根据步骤绘制程序流程图； 3、小组讨论需要用到的基本算法； 4、课堂练习，实现族谱录入； 5、组间互评，分析算法复杂度。	25min	**教学设计思路：** 1、课堂讨论，口述解决方案，帮助学生巩固树的特点，深化理论理解； 2、以任务为驱动，进行知识迁移与总结。程序的实现帮助学生在逻辑结构和功能实现之间建立连接，将理论知识转移为实践能力。此外，学生在实现程序过程中，将前面所学的关于树的运算的零散知识点汇聚在一起，构建起了完整的关于树的知识架构，提高了编程的灵活性； 3、知识拓展，通过对程序进行算法复杂度分析，提出其优劣，培养学生全方位考虑问题的能力。 **教学资源：** PPT 讲解， 示例图演示，编程环境。
课堂小结				
11	本节主要介绍了树和二叉树两种数据结构的定义以及它们的存储结构和运算定义。 树是以分支关系定义的层次结构，结构中的数据元素之间存在着“一对多”的关系。	回顾本节课的知识点，构建知识架构。	3min	**设计思路：**帮助学生构建一个完整的知识体系。解发学生更多更深的思考，完成拓

	树中除根结点没有前驱外，其它每个结点只有一个前驱；所有结点都有零个或多个后继。因此树型结构为自然界和计算机应用中出现的具有层次关系或分支关系的数据，提供了一种自然的表示方法。如可用树型结构描述人类社会的族谱和各种社会组织机构等。 在计算机学科和应用领域中，树也得到广泛应用。例如在编译程序中，可用树来表示源程序的语法结构等。树还是一种典型的递归结构。 二叉树是另一种树型结构，是度为 2 的有序树。本章重点讨论了二叉树。二叉树中每个结点至多有两个孩子，它有明确的左、右之分。 一棵深度为 h 的二叉树，最少含有 h 个结点，最多含有 2h-1 个结点；一棵具有 n 个结点的二叉树，其最小深度为 □log2n」+1。			展练习。
项目分析				
12	课堂连线企业导师，对之前利用线性结构、栈以及队列实现系统功能的项目实践进行点评，分析系统在当前状态下还需优化的方向。并分析本节课在项目优化中的地位和作用，明确下一步需要开发的内容和方向。	根据企业导师点评进行自我反思，及时进行自我学习调整，分析本节课项目实现的设计要点，进行算法设计。	8min	**教学设计思路：** 企业导师根据行业标准进行项目分析及点评，帮助学生及时发现问题、解决问题，强化对知识的理解，提升学生的职业素养。

八、知识拓展

1、拓展训练

项目名称：学生成绩管理系统项目

内容：对学生成绩进行量化考核，根据考核等级进行分类，利用树形结构将学生记录按照考核结果以层级式录入，方便后续查询。

2、考证（考研）训练

（2018 年考研真题）设一棵非空完全二叉树 T 的所有叶结点均位于同一层，且每个非叶结点都有 2 个子结点。若 T 有 k 个叶结点，则 T 的结点总数是（ ）。

A. 2k-1　　B. 2k　　C. k^2　　D. 2^k-1

（软件水平考证）对于一棵具有 n 个结点、度为 4 的树来说，树的高度至少是（ ）。

A. log4（2n）　　B. log4(3n- 1)　　C. log4(3n+ 1)　　D. log4 (2n+ 1)

3、竞赛训练

蓝桥杯二叉树部分 https://dasai.lanqiao.cn/notices/1096/

九、教学评价与反馈

本次树章节的教学内容涵盖了树的定义、基本操作等多个方面，从概念、原理、实现等多个角度深入浅出地介绍了二叉树的相关知识点。整个教学过程中，采用了举例、动画演示等多种教学方式，让学生能够更好地理解和掌握二叉树的相关知识。

针对本章的教学，我认为可以做出以下改进和优化：

1、更加注重实践：在讲解二叉树的实现原理和相关操作时，可以加强对实际场景的应用，让学生更加深入地理解二叉树的实际使用场景和应用价值；

2、增加练习：在讲解完二叉树的基本概念和操作后，可以适当多增加一些练习题，让学生能够更好地巩固所学知识点，并在实践中掌握二叉树的实际应用。

总的来说，本节的教学内容全面、系统，教学方法多样，但也存在一些可以进一步优化的地方，希望能够不断改进和完善，让学生更好地掌握二叉树的相关知识。

十、教学总结

二叉树是一种最重要的数据结构，二叉树有两种存储方式：顺序存储和链式存储。顺序存储结构适合满二叉树和完全二叉树，而链式存储结构适合一般的二叉树。二叉树和树的链式存储结构可采用动态链表，也可采用静态链表。对于链式存储结构，要掌握结点的结构及指针域的作用；对于顺序存储结构，要掌握用结点的位置关系来表示结点间父子关系的方法。

十一、项目实现代码（关键代码段）

```
/*家谱成员信息*/
typedef struct _tMember {
    char name[STR_LEN];       /*姓名*/
    char wife[STR_LEN];       /*妻子*/
}Member, * pMember;
/*家谱树节点*/
typedef struct _tTreeNode {
    Member member;                                  /*成员信息*/
    struct _tTreeNode* parent;                      /*父节点*/
    struct _tTreeNode* children[CHILD_LEN];         /*孩子节点*/
    int count;                                      /*孩子数量*/
    int level;                                      /*节点层级*/
}TreeNode, * pTreeNode;
/*创建树节点*/
pTreeNode createTreeNode(pTreeNode parent, pMember member) {
    pTreeNode node = (pTreeNode)calloc(1, sizeof(TreeNode));
    if (node) {
        node->member = *member;
        if (parent) {
            node->parent = parent;
            node->level = parent->level + 1;
```

```
            parent->children[parent->count++] = node;
        }    }
    return node;
}
/*删除树节点*/
void removeTreeNode(pTreeNode cursor) {
    if (cursor) {
        if (cursor->parent) {
            pTreeNode parent = cursor->parent;
            int position = -1;
            for (int index = 0; index < parent->count; ++index) {
                if (parent->children[index] == cursor) {
                    position = index;
                    break;
                }           }
            if (position != -1) {
       for (int index = position + 1; index < parent->count; ++index)
        {  parent->children[index - 1] = parent->children[index];
                }
                --parent->count;
            }        }    }}
/*计算成员层级*/
int countMemberLevel(const char content[]) {
    int level = 0;
    const char* cursor = content;
    while (*cursor == ' ') {
        ++level;
        ++cursor;
    }
    return level;
}
/*编辑家谱成员信息*/
void editFamilyMember(pMember member) {
    printf(" ╔----------------------------------------╗ \n");
    printf("              $ 编辑家谱成员信息 $\n");
    if (strlen(member->name)) {
        printf(" 姓名: %s\n", member->name);
    }
    else {
        printf(" 姓名: ");
        scanf("%s", member->name);
    }
    printf(" 妻子: ");
    scanf("%s", member->wife);
```

```
    printf("╚----------------------------------------╝\n");
}
/*添加家谱成员*/
void addFamilyTree(pTreeNode* root) {
    char name[STR_LEN] = { 0 };
    printf("╔----------------------------------------╗\n");
    printf("              $ 添加家谱成员 $\n");
    printf("  输入新成员姓名: ");
    scanf("%s", name);
    if (*root) {
        if (!recursiveFamilyTreeNodeFind(*root, name)) {
            pTreeNode target = NULL;
            char parentname[STR_LEN] = { 0 };
            printf("  输入新成员的父亲或者母亲名字(指定关系): ");
            scanf("%s", parentname);
            target = recursiveFamilyTreeNodeFind(*root, parentname);
            if (target) {
                Member member = { 0 };
                strcpy(member.name, name);
                editFamilyMember(&member);
                createTreeNode(target, &member);
                saveFamilyTree(*root);
                showFamilyMemberTitle();
                showFamilyMember(&member, 1);
                printf("----------------\n");
                printf("成功添加以上家谱成员! \n");
            }
            else {
                printf("添加失败, 家谱中未找到该名字! \n");
            }
        }
        else {
            printf("添加失败, 该成员名称已经存在! \n");
        }
    }
    else {
        Member member = { 0 };
        strcpy(member.name, name);
        editFamilyMember(&member);
        *root = createTreeNode(NULL, &member);
        saveFamilyTree(*root);
        showFamilyMemberTitle();
        showFamilyMember(&member, 1);
        printf("----------------\n");
```

```
        printf("成功添加以上家谱成员！\n");
    }
    printf("╚----------------------------------------╝\n");
}
/*删除家谱成员*/
void removeFamilyTree(pTreeNode* root) {
    pTreeNode target = NULL;
    char name[STR_LEN] = { 0 };
    printf("╔----------------------------------------╗\n");
    printf("              $ 删除家谱成员 $\n");
    printf("  输入姓名：");
    scanf("%s", name);
    target = recursiveFamilyTreeNodeFind(*root, name);
    if (target) {
        showFamilyMemberTitle();
        showFamilyMember(&target->member, 1);
        removeTreeNode(target);
        recursiveFamilyTreeNodeClear(target);
        if (target == *root) {
            *root = NULL;
        }
        saveFamilyTree(*root);
        printf("----------------\n");
        printf("成功删除以上家谱成员！\n");
    }
    else {
        printf("  没有找到相关信息！\n");
    }
    printf("╚----------------------------------------╝\n");
}
```

《数据结构及算法设计》教案 6，13-14 学时

一、教学基本信息			
课程名称	二叉树的遍历	**授课教师**	申妙芳
授课班级	21 级计算机应用工程 3 班	**授课时数**	2
授课时间	周二 3-4 节	**授课地点**	81107

二、教学分析	
教学内容	二叉树是一种常用的数据结构，它具有良好的可读性和高效的查找、插入、删除操作。在学习二叉树的遍历时，可以介绍前序遍历、中序遍历和后序遍历的概念和实现方法，并通过具体的例子进行讲解。
学情分析	本节课是在上节课的基础上对二叉树的实际应用，具有较高的实操性。因此，在学习本节课的过程中，要确保学生对二叉树的基本概念有较为清晰的认知，对于二叉树的结构形态建立了良好的空间想象，并对于前导课程《C 语言程序设计》中递归的应用掌握较为熟练，能够用递归的思想考虑问题。这些都需要纳入课前预习内容中。在本节课讲授过程中，出来结合虚拟仿真、图例演示、头脑风暴、组内讨论等形式外，也需要将“对比、复习”贯穿始终，引导学生将所学知识迁移至本节课，深化对遍历行为的理解

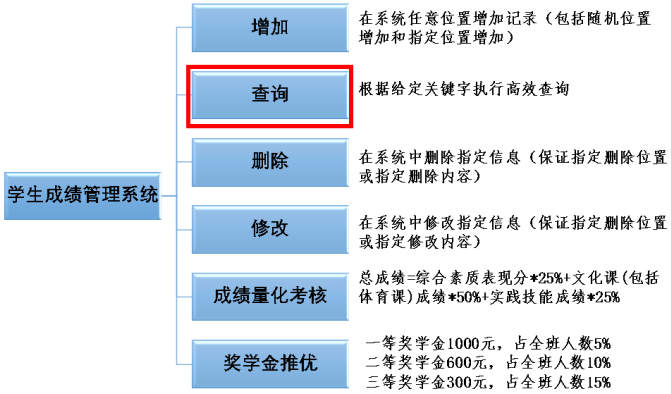
三、教学目标确定		
教学目标	**知识目标**	1、熟悉二叉树的遍历方式，包括前序遍历、中序遍历和后序遍历，能够实现二叉树的遍历； 2、能够根据具体问题，选择合适的遍历方式和数据结构，解决相应的问题。
	能力目标	1、分析和理解二叉树的数据结构，能够实现二叉树的基本操作； 2、掌握二叉树的遍历方式，理解遍历方式的实现原理和运用场景； 3、能够根据具体问题，选择合适的遍历方式和数据结构，解决相应的问题； 4、具备抽象思维能力，能够将二叉树的概念和方法应用到其他领域。
	素质目标	1、培养学生具备辩证思维的能力； 2、培养学生具有热爱科学、勇于实践、实事求是的学风和创新意识、创新精神； 3、培养学生具有较强的执行能力以及较高的工作效率和安全意识；

		4、培养学生规范、严谨、精确的工作态度和情感。
教学重点		教学重点是让学生掌握二叉树的遍历方式和线索二叉树的应用，培养学生的抽象思维能力和解决问题的能力。
教学难点		教学难点是让学生理解线索二叉树的概念和实现原理，以及在解决实际问题时选择合适的遍历方式和数据结构。

四、思政融入课堂

1、强化学生的责任感和使命感。教育学生要树立正确的人生观和价值观，认识到自己在学习和实践中所承担的责任和使命。在学习二叉树和线索二叉树的过程中，教育学生应该具备坚持不懈、细心认真、耐心研究的品质，以达到掌握这些知识和技能的目标；
2、培养学生的创新能力。二叉树是计算机科学中常用的数据结构之一，而计算机科学本身也是一门不断发展、变革的科学。在学习这些知识和技能的过程中，教育学生要积极探索、创新，不断探索新的应用场景和解决方法，培养学生的创新思维能力；
3、强调学生的团队协作精神。在计算机科学领域中，团队协作是非常重要的能力，也是计算机科学家必备的素质之一。在学习二叉树的过程中，教育学生要重视团队协作，培养学生的团队精神和协作能力，让学生学会与人合作，共同完成一项任务。

五、课程在项目中的定位



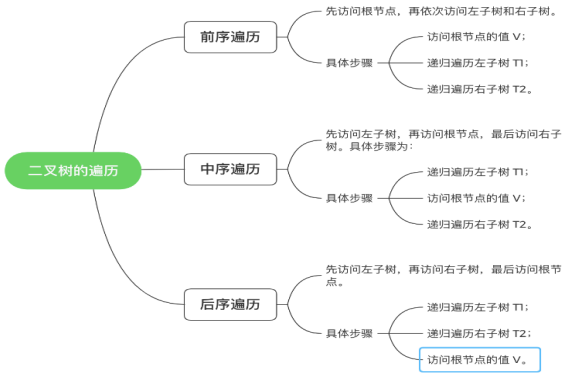
拓展“查询”功能，利用树形结构按照考核等级以层级形式打印输出学生信息组织结构图。

六、教学策略

设计思路	1、具体思路： 遍历是二叉树上最重要的运算之一，是二叉树上进行其它运算之基础。由于访问结点所做的操作依赖于具体的应用问题，所以我们对所讲解的问题分别举了一个容易和较难的例子，并配上一个难度适中的启发式的习题思考，在讲解过程中让学生充分理解，并学会运用不同的遍历方式解决问题，最后加上顺口溜式的总结，使学生对知识点更加融会贯通。 2、教学方法：

抓住学生的注意力激发和维持学习动机。在二叉树的讲解中贯穿一条主线，即逻辑结构，存储结构，如何通过算法实现基本运算。二叉树的结构是非线性的，这些内容借助于多媒体教学，以生动、形象，灵活的方式激发学生的学习兴趣，通过多个感官司来获取信息。

3、思维导图：



教学资源

学习环境：机房、局域网，交互式电子黑白。

学习资源：

1、专题学习网站：包括慕课网、学习通、职教云等；

2、授课课件：根据此节学习内容制作的 PPT 课件；

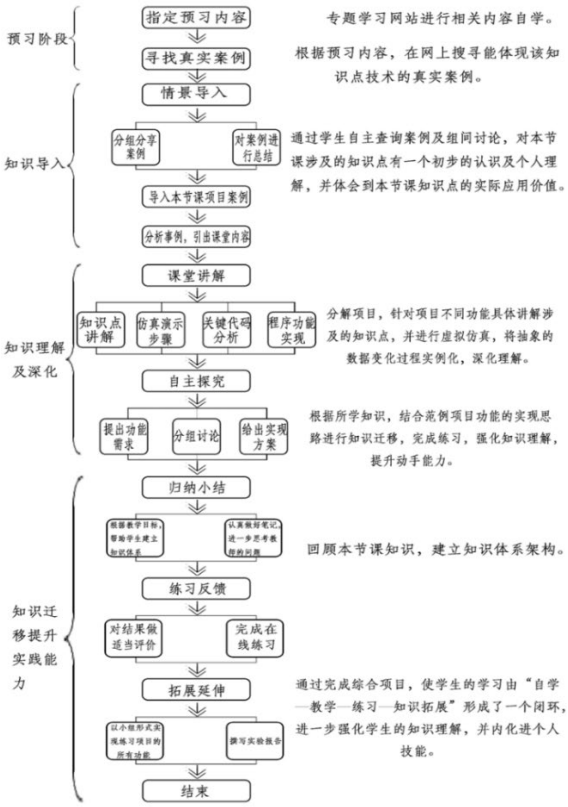
3、多媒体资源库：虚拟仿真演示案例、项目演示案例；

4、案例库：课堂练习题库、测验题库等；

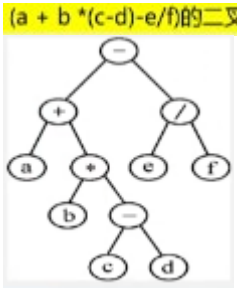
5、精品课程网站：本门课程的精品课程网站；

6、VC6.0++运行环境：进行案例演示及学生练习项目的专业运行环境。

教学流程

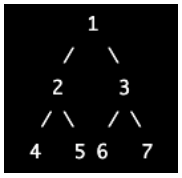


七、教学过程

教学环节	教学活动		时间安排	设计思路与教学手段
	教师授课内容	学生		
课前预习				
1	1、提供慕课网资源，请同学们预习本次课程基本知识，如二叉树的定义，性质、存储结构、递归等； 2、思政学习：AI 医疗影像行业发展 提出问题：二叉树的应用场景有哪些？	1、在充足的时间内观看慕课视频； 2、完成精品课成基础练习； 3、分组收集二叉树的应用案例。	课前一周	**设计思路：**利用慕课网自主学习，提高学生学习主动性和创造性，强化解决问题的能力。 **涉及教学资源：** 慕课网、搜索引擎。
知识导入				
2	**回顾上节课内容：**学习了二叉树的性质，二叉树的存储结构，有顺序存储结构和链式存储结构； **分析学生案例，**探讨二叉树在现实生活中的应用。	以组为单位分享二叉树的实际应用案例。	3min	**教学手段：** 分组讨论、真实案例。
3	**引出课堂案例：** 大部分算术运算符均有两个操作数，所以一般可以用二叉树来表示一个算术表达式。 	观察案例，组内讨论如何用二叉树来解决表达式求值。	2min	**设计思路：**激发学生的学习兴趣和积极性
4	**分析案例，引出新课：** 若表达式为常数或简单变量，则相应的二叉树为仅有一个根结点的二叉树，其数据域存放该表达式的信息。 若表达式为“第一操作数、运算符、	观察案例，思考二叉树的其它应用案例。	2min	**设计思路：**引导学生学习，发挥学生主观能动性。

	第二操作数”的形式，则相应二叉树的左子树表示第一操作数，右子树表示第二操作数，根结点的数据域存放运算符。 若运算符为一元运算符，则左子树为空，右子树表示操作数，根结点的数据域存放运算符。 **提问学生：**表达式中的括号怎样表示了？			
知识讲解				
5	**板书重要知识** **1、遍历二叉树的定义** 遍历二叉树是指按照某种顺序依次访问二叉树中的每个节点，使每个节点都被访问一次且仅访问一次的过程。常用的二叉树遍历方式有前序遍历、中序遍历和后序遍历。 **2、分别用动画演示二叉树三种遍历过程** **3、分别用图例分析三种遍历的行为特点** **4、引导学生用递归的思想总结遍历步骤**	1、了解遍历的含义 2、通过观看仿真动画，讨论总结三种遍历的步骤 3、尝试用递归的思想进行分析	10min	**设计思路：** 1、仿真演示。帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、课堂讨论，头脑风暴。引导学生思考遍历的行为，总结遍历的变化规律和特点，并尝试编写代码段。这个过程可以帮助学生进一步内化抽象知识，在空间想象和实践之间建立联系；培养学生的主观能动性和创新能力。 **教学资源：** PPT 讲解，示例图演示，案例分析，虚拟仿真。
6	**教师巡堂，个性化辅导** **知识深化：** 1、课堂理论练习 登录职教云平台完成理论练习。 **2、动画演示二叉树三种遍历过程** **总结遍历规则：** 口诀： DLR—先序遍历，即先根再左再右； LDR—中序遍历，即先左再根再右； LRD—后序遍历，即先左再右再根。 **3、根据三种遍历的递归步骤画出程序流程图，并写出相应的代码段**	1、完成职教云课堂练习 2、复习回忆 3 中遍历的步骤，组内讨论画出程序流程图，并写出相应的代码段	20min	**设计思路：** 遍历理论理解方面不难，但是将理论理解应用于程序实践，这个转换过程非常的抽象，需要学生在理论理解透彻的基础上，建立非常清晰的空间想象，才能够顺利完成逻辑到实践 的转换。 1、课堂练习，巩固理

以以下二叉树为例：



其中，节点的编号表示节点的值。

前序遍历（Pre-order traversal）：先访问根节点，然后访问左子树，最后访问右子树。对于上述二叉树，前序遍历的顺序是：1，2，4，5，3，6，7。

```
public static void preOrderTraversal(TreeNode head) {
    if (head == null) {
        return;
    }
    System.out.print(head.val + " ");
preOrderTraversal(head.left);
preOrderTraversal(head.right);
}
```

中序遍历（In-order traversal）：先访问左子树，然后访问根节点，最后访问右子树。对于上述二叉树，中序遍历的顺序是：4，2，5，1，6，3，7。

```
public static void inOrderTraversal(TreeNode head) {
    if (head == null) {
        return;
    }
    inOrderTraversal(head.left);
    System.out.print(head.val + " ");
    inOrderTraversal(head.right);
}
```

后序遍历（Post-order traversal）：先访问左子树，然后访问右子树，最后访问根节点。对于上述二叉树，后序遍历的顺序是：4，5，2，6，7，3，1。

需要注意的是，不同的遍历方式会得到不同的节点访问顺序，但是每个节点都会被访问且仅访问一次。

```
public static void postOrderTraversal(TreeNode head) {
    if (head == null) {
```

论理解。现学现练，有助于巩固学生的理论知识，强化理解

2、虚拟仿真。帮助学生建立空间想象，深化理论知识

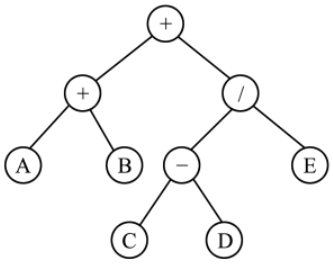
3、程序实践。在建立了空间想象的基础上用程序的语言进行遍历的行为描述，完成从逻辑到实践的转换过程，内核理论，提升实践能力。

培养学生的团队合作精神和团队意识。

教学资源：

PPT 讲解，示例图演示，案例分析，虚拟仿真、编程环境。

	return; } postOrderTraversal(head.left); postOrderTraversal(head.right); System.out.print(head.val + " "); }			
7	**知识拓展** **先序遍历二叉树的非递归实现** 从根开始，当前结点存在或栈不为空，重复如下两步操作。 (1)访问当前结点，当前结点进栈，进入其左子树，重复直至当前结点为空； (2)若栈非空，则退栈顶结点，并进入其右子树。 **现场示范实现代码：** void PreOrder(BiTree root) {SeqStack *S; BiTree p; InitStack(S); p＝root; while(p! ＝NULL||! IsEmpty(S)) {while(p! ＝NULL) {Visit(p->data);Push(S,p);p＝p->LChild;} if(! IsEmpty(S)) {Pop(S , &p);p ＝ p->RChild;} } } **中序遍历二叉树的非递归实现：** 从根开始,当前结点存在或栈不为空，重复如下两步操作。 (1)当前结点进栈，进入其左子树，重复直至当前结点为空； (2)若栈非空，则退栈，访问出栈结点，并进入其右子树。 **现场示范实现代码：** void InOrder1(BiTree root) {SeqStack *S; BiTree p; InitStack(S);p＝root; while(p! ＝NULL||! IsEmpty(S)) {while(p! ＝NULL)	1、升华课前知识； 2、吸收课中难点； 3 积极回答问题。	18min	**设计思路：**1、对比与联系。分别用递归的思想和非递归思想描述同一行为，更有助于学生理解，并逐步完善遍历的整体知识架构； 2、头脑风暴。这个过程可以帮助学生进一步内化抽象知识，在空间想象和实践之间建立联系； 3、思维拓展。同样的问题用不同的方法解决，培养学生在解决问题时进行多方面思考的能力和习惯。 **教学资源：** PPT 讲解，示例图演示，案例分析，编程环境。

	{Push(S, p);p=p->LChild;} if(! IsEmpty(S)) {Pop(S , &p);Visit(p->data);p=p->RChild;} } }			
8	**项目实践** **1、项目分析** $(A+B)+(C-D)/E$  运算符栈 OPTR，来暂存已经扫描到的还未处理的运算符。 每两个操作数和一个运算符就可以建立一棵表达式二叉树，而该二叉树又可以作为另一个运算符结点的一棵子树。 表达式树栈 EXPT，来暂存已建立好的表达式树的根结点，以便其作为另一个运算符结点的子树而被引用。 2、**教师巡堂，个性化辅导。**	1、小组讨论具体实现步骤，并根据步骤绘制程序流程图； 3、小组讨论需要用到的基本算法； 4、课堂练习，实现表达式求值； 5、组间互评，分析算法复杂度。	25min	**教学设计思路：** 1、课堂讨论，口述表达式求值过程，帮助学生巩固二叉树遍历的特点，深化理论理解； 2、以任务为驱动，进行知识迁移与总结。程序的实现帮助学生在逻辑结构和功能实现之间建立连接，将理论知识转移为实践能力。此外，学生在实现程序过程中，将前面所学的关于二叉树遍历的运算的零散知识点汇聚在一起，构建起了完整的关于二叉树的知识架构，提高了编程的灵活性； 3、知识拓展，通过对程序进行算法复杂度分析，提出其优劣，培养学生全方位考虑问题的能力。 **教学资源：** PPT 讲解， 示例图演示，编程环境。
课堂小结				
9	遍历二叉树的运算内容包括：前序遍历、中序遍历、后序遍历、层序遍历。这些遍历方式是对二叉树进行访问的基本方法，它们可以用递归或非递归的方式实	回顾本节课的知识点，构建知识架构。	3min	**设计思路：**帮助学生构建一个完整的知识体系。解发学生更多更深的思考，完成拓

	现。			展练习。
项目分析				
10	课堂连线企业导师，对上节课利用栈实现系统功能的项目实践进行点评；对本节课利用队列实现系统功能进行项目分析，说明评价标准，强调注意事项。	根据企业导师点评进行自我反思，及时进行自我学习调整，分析本节课项目实现的设计要点，进行算法设计。	8min	**教学设计思路：** 企业导师根据行业标准进行项目分析及点评，帮助学生及时发现问题、解决问题，强化对知识的理解，提升学生的职业素养。

八、知识拓展

1、拓展训练

项目名称：学生成绩管理系统项目

内容：对学生成绩进行量化考核，根据考核等级进行分类，利用树形结构将学生记录按照考核结果进行分类录入，通过遍历的形式以树形形式执行打印输出，方便查看。

2、考证（考研）训练

（2012 年考研真题）若一棵二叉树的前序遍历序列为 a,c,b,d,c,后序遍历序列为 b，c，d，e，a，则根结点的孩子结点（　　）。

A　只有 e　　B. 有 e、b　　C. 有 e、c　　D. 无法确定

（软件水平考试）某二叉树的前序和后序遍历序列正好相反，则该二叉树一定是（）（4 分）

A　空或只有一个结点

B　高度等于其结点数

C　任一结点无左孩子

D　任一结点无右孩子

3、竞赛训练

蓝桥杯二叉树遍历部分 https://dasai.lanqiao.cn/notices/1096/。

九、教学评价与反馈

授课过程中，思路清晰，语言生动而富于亲和力，表达多样，激发学生兴趣，幻灯片布局简洁，动画紧随讲解节奏的推进而随时跟进，生动形象，而又清晰明了地将所讲的知识点呈现出来。在视频讲解中，通过改变教学内容的展现形式，更具独到趣味，激发学生内外动力来实现。在今后教学中，我们可以通过更多提问等环节，加入与学生的互动，引导其思考，让学生更加充分融入到学习之中。

十、教学总结

本次教学内容主要涵盖了遍历二叉树的相关知识，其中遍历二叉树部分介绍了四种遍历方式的概念、实现方法以及应用场景。

在教学中，应重点强调遍历二叉树的四种遍历方式以及其递归和非递归实现方式，通过实例演示和练习来帮助学生掌握基本的遍历技巧，并引导学生思考如何根据不同应用场景选择合适的遍历方式。

最后，教学过程中还应引导学生思考二叉树的应用场景和实际问题，通过案例分析和实例演示来激发学生对数据结构的兴趣和创造力。

十一、项目实现代码（关键代码段）

```
    //打印所有学生的信息(中序遍历打印)
    void tree_for_each(student_node *root)
    {
       if (root == NULL)
       {
          return;
       }
       tree_for_each(root->lchild);
       printf("编号: %d,学生姓名: %s,学生成
绩: %d\n",root->perfor_info.number,root->
    perfor_info.stu_name,root->perfor_info.grade);
       tree_for_each(root->rchild);
    }
    //删除学生成绩信息
    student_node *remove_tree_node(student_node *root, int rm_value)
    {
        student_node *tmp;
        if(root == NULL)//如果树为 NULL 则直接返回
            return NULL;
        if(root->perfor_info.number == rm_value)
        {
            if(root->lchild==NULL && root->rchild == NULL)
            {
                free(root);
                return NULL;
            }
            else if(root->lchild != NULL)//如果有左子树
            {
                for(tmp=root->lchild; tmp->rchild != NULL;
     tmp=tmp->rchild);//用 tmp 来记录 root 的左子树当中的最右边的节点
    (也就是左树当中的最大值)
                root->perfor_info.number = tmp->perfor
    _info.number;//将左子树中的最大值赋值到我们要删除的节点上（
    等同于覆盖了这个节点）
```

```
            root->lchild = remove_tree_node(root->
lchild, tmp->perfor_info.number);//递归的删除掉重复出来的这个节点,
将左子树的数据更新进来
        }
        else//如果只有右子树
        {
            for(tmp=root->rchild; tmp->lchild != NULL;
tmp=tmp->lchild);//用 tcmp 来记录 root 的右子树中的最左边的节点
（也就是有树当中的最小值）
            root->perfor_info.number = tmp->perfor_info.number;
//将右子树中的最小值赋值到我们要删除的节点上（等同于覆盖了这个节点）
            root->rchild = remove_tree_node(root->rchild, tmp->
perfor_info.number);//递归的删除掉重复出来的这个节点,
将右子树的数据更新进来
        }
    }
    else if(rm_value < root->perfor_info.number)
//如果删除的数据比这个节点要小，则继续往左边去删除节点
    {
        root->lchild = remove_tree_node(root->lchild, rm_value);
    }
    else//如果删除的数据比这个节点要大，则继续往右边去删除
    {
        root->rchild = remove_tree_node(root->rchild, rm_value);
    }
    //插入新节点结束后再判断是否出现不平衡
    if(get_tree_height(root->lchild)-get_tree_height
(root->rchild) == 2)//左不平衡
    {
        //如果插入的数据比跟的左树节点的数据要小,
那他肯定是插入到 root->lchild 的左边去，出现了左左不平衡
        if( get_tree_height(root->lchild->lchild)-get_tree_height
(root->lchild->rchild) == 1)//左左不平衡
        {
            root = tree_node_rotate_right(root);
        }
        else//否则则是插入到 root->lchild 的右边去,
出现了左右不平衡
        {
            root = tree_node_rotate_left_right(root);
        }
    }
    else if(get_tree_height(root->rchild)-get_tree_height
(root->lchild) == 2)//右不平衡
```

```
    {
        if( get_tree_height(root->rchild->rchild)-get_tree_height
(root->rchild->lchild) == 1)//右右不平衡
        {
            root = tree_node_rotate_left(root);
        }
        else//右左不平衡
        {
            root = tree_node_rotate_right_left(root);
        }
    }
    root->height = get_tree_height(root);
    return root;//如果当前 root 这个节点不是我们删除的节点,
我们便原封不动的返回出去
}
int main(void)
{
   int choose,input_value;
   struct stu_grade input_student_data;//定义一个学生结构体类
型的数据用来缓存学生数据
   student_node *root = NULL,*new_node,*find_node;
   //初始化一个节点 root 为空
   while(1)
   {
      printf("*****************欢迎来到学生成绩管理系统********\n");
      printf("请输入数字选择相应的指令\n");
      printf("1、增加插入学生的成绩\n");
      printf("2、搜索学生成绩\n");
      printf("3、打印所有学生的成绩\n");
      printf("4、删除学生的成绩\n");
      printf("5、退出学生成绩管理系统\n");
      printf("*****************************************************\n");
      scanf("%d",&choose);
      switch(choose)
      {
         case 1:
            printf("请您输入学生的编号: \n");
            scanf("%d",&input_student_data.number);
            printf("请您输入学生的姓名: \n");
            scanf("%s",input_student_data.stu_name);
            printf("请您输入学生的成绩: \n");
            scanf("%d",&input_student_data.grade);
            new_node = request_student_node(&input_student_data);
            root = insert_node_to_tree(root,new_node);
```

```
                break;
            case 2:
                printf("请输入学生的编号来查询学生成绩信息\n");
                scanf("%d",&input_value);
                find_node = find_tree_node(root,input_value);
                if (find_node == NULL)
                {
                    printf("找不到该学生的信息\n");
                    break;
                }
                printf("编号：%d,学生姓名：%s,学生成
绩：%d\n",find_node->perfor_info.number,find_node->
    perfor_info.stu_name,find_node->perfor_info.grade);
                break;
            case 3:
                printf("所有的学生的成绩信息为\n");
                tree_for_each(root);
                break;
            case 4:
                printf("请输入学生的编号来删除学生成绩信息\n");
                scanf("%d",&input_value);
                root = remove_tree_node(root,input_value);
                break;
            case 5:
                 goto log_out;
        }
    }
    return 0;
log_out:
    return 0;
}
```

《数据结构及算法设计》教案 7，15-16 学时

一、教学基本信息			
课程名称	树表查找	**授课教师**	申妙芳
授课班级	21 级计算机应用工程 3 班	**授课时数**	2
授课时间	周一 7-8 节	**授课地点**	81107

二、教学分析	
教学内容	树表查询是数据库中常用的一种数据结构，它能够高效地支持数据的插入、删除和查找操作。其中，二叉排序树、平衡二叉树和 B 树都是树表查询中的重要概念。在教学树表查询时，需要重点介绍二叉排序树、平衡二叉树和 B 树的原理、实现方式、时间复杂度等内容，并通过实例讲解它们在实际应用中的场景和优缺点。同时，还需要进行大量的实践训练，让学生掌握如何使用这些数据结构来解决实际问题。
学情分析	学习树表查询需要具备一定的数据结构和算法基础，能够理解二叉树、平衡二叉树、B 树等概念，并能够进行代码实现和调试。同时，还需要具备一定的计算机操作系统和磁盘文件系统的基础知识，能够理解 B 树在磁盘文件系统中的应用场景和优势。对于不同程度的学生，需要分别采用不同的教学方法和策略，以便更好地提升他们的学习兴趣和成果。并且需要通过练习来加深对树表查找的理解。

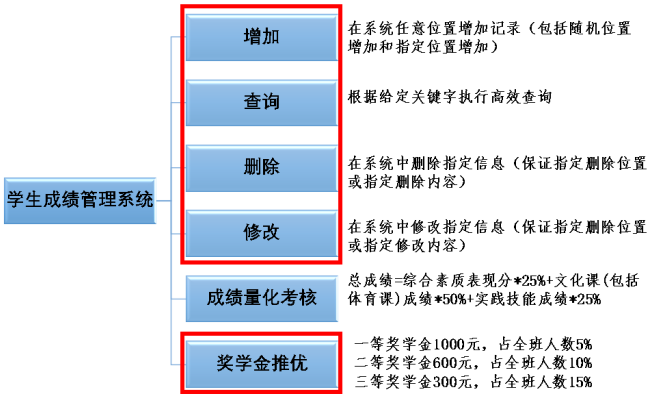
三、教学目标确定		
教学目标	**知识目标**	1、理解树表查询的概念、特点和原理，包括二叉排序树、平衡二叉树和 B 树； 2、掌握树表查询的实现方式和算法，能够进行代码实现和调试； 3、了解树表查询在实际应用中的场景和优缺点，能够根据需求选择合适的数据结构进行操作。
	能力目标	1、能够运用树表查询来解决实际问题，如数据插入、删除、查找等操作； 2、能够分析和评估不同数据结构的优劣，选择合适的数据结构进行操作，提高算法的效率； 3、能够对算法进行优化和改进，提高程序的运行效率和稳定性。
	素质目标	1、具有良好的思想品德和诚实、敬业、负责等职业道德； 2、具有良好的文化修养； 3、具有良好的团结协作精神、团队意识、组织协调能力。

教学重点	1、各种查找方法所需要的存储结构及各种查找方法的优缺点； 2、各种查找方法在等概率情况下，其平均查找长度的计算方法； 3、二叉排序树的建立、查找、插入和删除运算的算法实现。
教学难点	二叉排序树上删除结点的运算。

四、思政融入课堂

在二叉排序树查找中给学生传达“提高效率”的思想，引导学生在解决复杂工程问题时尽量以节省时间和空间成本为目标来设计解决方案，将“统筹规划、大局意识”融入到课程难点讲授中。

五、课程在项目中的定位

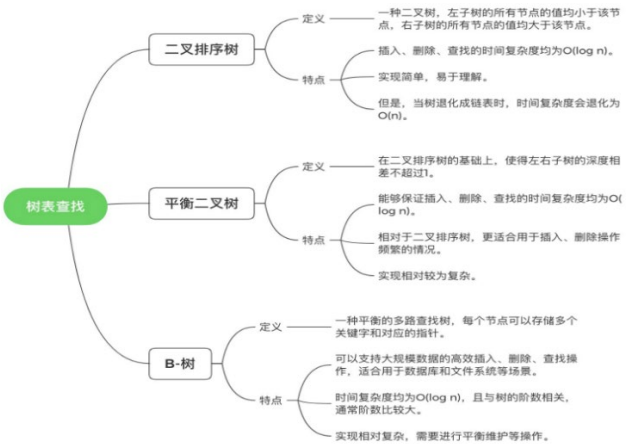


利用树表查找思想优化查找功能算法，实现个性、高效的查找。从而提高“增、删、查、改、奖学金推优”功能中与查找相关的程序运行效率，对系统进行综合优化。

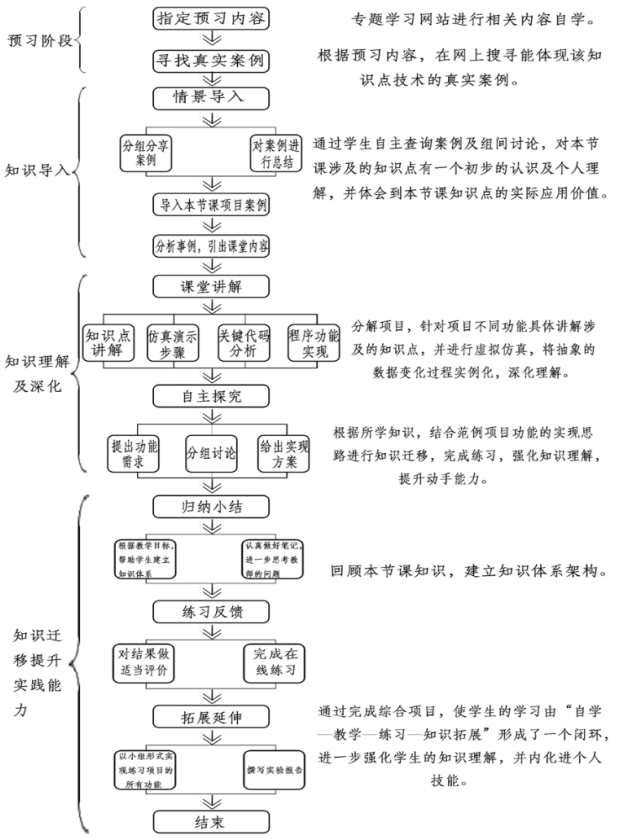
六、教学策略

设计思路

1、思维导图：



2、具体思路：

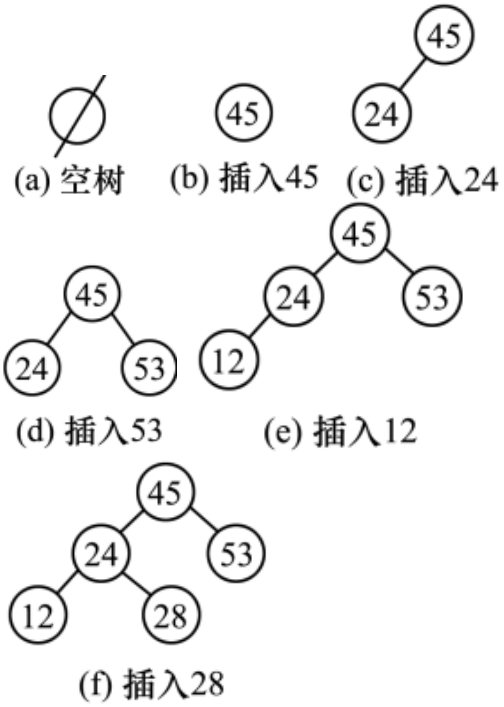
	树表查找是利用特定的树结构将查找表组织成有序表，并在表上实现查找、插入和删除运算。基于树结构的查找表本身是在查找过程中动态生成的。查找的算法很多，根据数据集合的不同特点使用不同的查找算法，可以节省空间与时间，提高程序效率。本节将重点围绕查找算法的原理与代码实现进行介绍。在二叉排序树查找中给学生传达“提高效率”的思想，引导学生在解决复杂工程问题时尽量以节省时间和空间成本为目标来设计解决方案，将“统筹规划、大局意识”融入到课程难点讲授中。 3、教学方法： 采用 PBL 问题解决式教学法。为学生提供教材导读、学习任务、情景案例，以小组形式展开课前学习，以提出问题为目标，培养学生思维能力。教学中，实践项目教学安排提倡学习小组围坐，教师巡回学习小组之中;教师下达任务书，学生围绕填报任务书为目的，自主探究与讨论;最终，教师根据任务书，给予不同小组辅助问题解答以及个性化的指导，完成授课环节。课后，引入小助教（学委、学习组长）机制。遵循当代大学生的心理特征，助教机制有效打破老师和学生之间的沟通障碍。
教学资源	**学习环境：**机房、局域网，交互式电子黑白。 **学习资源：** 1、专题学习网站：包括慕课网、学习通、职教云等； 2、授课课件：根据此节学习内容制作的 PPT 课件； 3、多媒体资源库：虚拟仿真演示案例、项目演示案例； 4、案例库：课堂练习题库、测验题库等； 5、精品课程网站：本门课程的精品课程网站； 6、VC6.0++运行环境：进行案例演示及学生练习项目的专业运行环境。
教学流程	

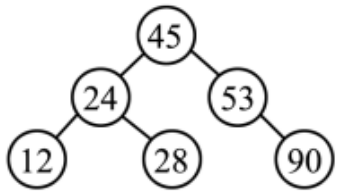
七、教学过程

教学环节	教学活动		时间安排	设计思路与教学手段
	教师授课内容	学生		
课前预习				
1	1、提供慕课网资源，请同学们预习本次课程基本知识，如查找的基本概念，常用查找算法等； 2、思政学习：自动驾驶技术 https://new.qq.com/rain/a/20230404A02P7M00 3、提出问题：对比各查找算法的优缺点？	1、在充足的时间内观看慕课视频； 2、完成精品课成基础练习； 3、分组收集查找算法。	课前一周	**设计思路：**利用慕课网自主学习，提高学生学习主动性和创造性，强化解决问题的能力。 **涉及教学资源：** 慕课网、搜索引擎。
知识导入				
2	**回顾上节课内容：** 1、二叉树的遍历方法； 2、简单补充学生收集到的查找算法；	以组为单位分享查找算法。	3min	**教学手段：** 分组讨论、真实案例。
3	**引出课堂案例：** 在学生成绩管理系统中，每年有新入学的学生信息，有转专业的学生信息，有毕业的学生信息，我们如何在不断变化的学生信息中快速查找出指定的学生的成绩？	观察案例，思考如何用树表查找来解决。	2min	**设计思路：**激发学生的学习兴趣和积极性。
4	**分析案例，引出新课：** 学生信息是动态的，采用树表查找法。利用特定的树结构将查找表组织成有序表，并在表上实现查找、插入和删除运算。基于树结构的查找 表本身就是在查找过程中动态生成的。 **给出其它应用案例：** 在学生管理系统中，查找某个学生，查找某个班主任等等。	观察案例，思考用自己收集的查找算法，效率会怎么样？	2min	**设计思路：**引导学生学习，发挥学生主观能动性。
知识讲解				

5	**板书重要知识：** **查找的基本概念** 查找表是一种以集合为逻辑结构，以查找运算为基本操作的数据结构.下面给出有关查找的基本概念。 集合:具有相同类型的数据元素(或记录)构成的整体,被称为集合。同集合中的各个数据元素互不相同,并且集合中的数据元素即结点之间不存在任何逻辑关系,但结点之间有可能存在非逻辑关系(数值大小、排列次序等)。 **动画仿真常用查找算法—折半查找** 基本思想： 1、设 R[low…high]，中点位置 mid=(low+high)/2；然后将待查的 K 值与 R[mid].key 比较，若相等，则查找成功并返回此位置； 2、若 R[mid].key>K，则该结点必定在 mid 左边的子表 R[1…mid-1]中； 3、若 R[mid].key<K，则该结点必定在 mid 右边的子表 R[mid+1…n]中。 **举例** 在以下 11 个数据元素的有序表中：(6,12,15,18,22,25,28,35,46,58,60)查找关键字(关键字即为数据元素的值)为 12、50 的数据元素。 **现场示范：** `int BinSearch(SqList L, KeyType k)` `{ low=1;high=L.lengh;` `while(low<=high)` `{ mid =(low+high)/2;` `if(k==L.item[mid].key) return (mid);` `else if(k<L.item[mid].key) high=mid-1;` `else low=mid+1;` `}` `return (0);` `}`	1、观看仿真动画，讨论折半查找的特点，总结步骤 2、根据步骤编写程序代码段	15min	**设计思路：** 1、仿真演示。帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、课堂讨论，头脑风暴。引导学生思考折半查找的规律和特点，并尝试编写代码段。这个过程可以帮助学生进一步内化抽象知识，在空间想象和实践之间建立联系； 培养学生的主观能动性和创新能力 **教学资源：** PPT 讲解，案例分析，虚拟仿真互动。
6	**教师巡堂，个性化辅导** **拓展训练** 顺序表中关键字序列为 6、13、	1、上机练习； 2、互助学习； 3、小组讨论。	10min	**设计思路：**培养学生的团队合作精神和团队意识。

	20、37，利用二分法查找 K=22 的位置，实现完整的查找过程，n=5。进行算法设计。			
7	**二叉排序树的性质** 1、若它的左子树非空，则左子树上所有结点的值均小于根结点的值； 2、若它的右子树非空，则右子树上所有结点的值均大于根结点的值； 3、左、右子树本身又各是一棵二叉排序树。 **请学生总结特点：** 任何一个结点必须大于它的左孩子，且小于它的右孩子。 **二叉排序树的查找** **用动画演示查找过程：** 根据二叉排序树的结构特点,二叉排序树可看作一个有序表,所以在二叉排序树上进行查找与折半查找类似。当二叉排序树非空时,首先将给定关键字 K 与根结点关键字 key 进行比较,若 K＝key,则查找成功,返回根结点的地址;若 K<key,则进一步查找左子树;若 K>key,则进一步查找右子树。通常采用二叉链表作为二叉排序树的存储结构,二叉排序树查找的递归算法。 **思政学习：** 二叉排序树查找中给学生传达“提高效率”的思想，引导学生在解决复杂工程问题时尽量以节省时间和空间成本为目标来设计解决方案，将“统筹规划、大局意识”。 **平均查找长度** 每个结点查找成功需要比较的次数总和，再除以结点个数。 **现场示范：** 二叉排序树查找代码实现： `if(T==NULL||key==T->key)` `return T;` `if(key<T->key)` `return SearchBST(T->lchild,key);` `else` `return SearchBST(T->rchild,key);`	1、观看仿真动画，讨论二叉排序树查找的特点，总结步骤 2、根据步骤编写程序代码段	15min	**设计思路：** 1、仿真演示。帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、课堂讨论，头脑风暴。引导学生思考二叉排序树查找的规律和特点，并尝试编写代码段。这个过程可以帮助学生进一步内化抽象知识，在空间想象和实践之间建立联系； 培养学生的主观能动性和创新能力 **教学资源：** PPT 讲解，案例分析，虚拟仿真

8	**教师巡堂，个性化辅导** **知识拓展：** 从具有n个结点的二叉排序树中查找一个元素时，在最坏情况下的时间复杂度为（ ）。	1、上机练习； 2、互助学习； 3、小组讨论。	5min	**设计思路：**培养学生的团队合作精神和团队意识。
9	**二叉排序树的插入原则** **用动画演示查找过程：** 二叉排序树是一种动态查找表。插入的原则是:若二叉排序树为空,则插入结点应为新的根结点;否则依据二叉排序树定义,在其左子树或右子树上查找自己的位置,当插入结点的值与树中某结点值相等时,则返回;否则若插入结点的值小于树中某结点值时,则继续在其左子树查找自己的位置,若插入结点的值大于树中某结点值时,则继续在其右子树查找自己的位置,直至某个结点的左子树或右子树为空树,表明给定值的结点在该二叉排序树中不存在,则新插入的结点应作为一个新添加的叶子结点并成为查找不成功的路径上最后一个结点的左孩子或 右 孩 子。 **举例：** 设关键字的输入顺序为:45,24,53,12,28,90,按上述算法生成二叉排序树的过程  	1、观看仿真动画，讨论二叉排序树插入的特点，总结步骤 2、根据步骤编写程序代码段	10min	**设计思路：** 1、仿真演示。帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、课堂讨论，头脑风暴。引导学生思考二叉排序树插入的规律和特点，并尝试编写代码段。这个过程可以帮助学生进一步内化抽象知识，在空间想象和实践之间建立联系； 培养学生的主观能动性和创新能力 **教学资源：** PPT 讲解，案例分析，虚拟仿真

	 (g) 插入90 **现场示范：** 二叉排序树插入代码实现： If(T==Null) { S=(BSTree *)malloc(sizeof(BSTNODE)); S->key=K; S->lchild=null; S->rchild=null; T=s; } else if(K<T->key) InsertBST(T->lchild,k); else if(K>T->key) InsertBST(T->rchild,k);			
10	**教师巡堂，个性化辅导** **知识拓展：** 假设有 10 个关键字，它们具有相同的 Hash 函数值，用线性探测法把这 10 个关键字存入 Hash 地址空间中，至少需要做(　)次探测。	1、上机练习； 2、互助学习； 3、小组讨论。	5min	**设计思路：**培养学生的动手实践能力，达到知识升华。
11	**平衡二叉树** 平衡二叉树又称为 AVL 树,它是一棵“平衡化”的二叉排序树。一棵平衡二叉树或者是空树,或者是具有下列性质的二叉排序树:①左子树与右子树的高度之差的绝对值小于等于 1;②左子树和右子树也是平衡二叉排序树。引入平衡二叉排序树的目的是为了克服二叉排序树中左、右子树深度不平衡的状态,从而提高查找效率,保持其平均查找长度为 log2n。 **失衡二叉树的调整：** **用动画演示查找过程：** 失衡 LL 型、LR 型、RR 型、RL 型利用旋转的方法将处于失衡状态的二叉	观看仿真动画，讨论二叉排序树插入的特点，总结步骤	5min	**设计思路：** 1、仿真演示。帮助学生建立抽象的空间想象，直观感受到地址的动态变换过程； 2、课堂讨论，头脑风暴。引导学生思考平衡二叉树查找的规律和特点； 培养学生的主观能动性和创新能力 **教学资源：** PPT 讲解，案例分析，虚拟仿真

	排序树调整至平衡状态,能够保持二叉排序树的有序特性,这一点可以利用中序遍历所得的关键字序列的有序性加以证明。因此当平衡二叉树因插入结点而失去平衡时(无论哪一种情况),仅需对最小不平衡子树进行平衡旋转处理,经过旋转处理恢复平衡的子树深度与插入之前的深度相等,不会影响插入路径上所有祖先结点的平衡度。			
12	**项目分析:** 1、引导学生口述查找过程; 2、将口语化的查找转化为流程图; 3、根据流程图转化为代码。  **教师巡堂,个性化辅导。**	1、总结查找步骤; 2、根据步骤绘制程序流程图; 3、小组讨论需要用到的基本算法; 4、课堂练习,实现项目功能; 5、组间互评,分析算法复杂度。	17min	**设计思路:**培养学生的团队合作精神和团队意识。
课堂小结				
13	1、熟练掌握线性表的三种查找方法:顺序查找、折半查找和分块查找并能灵活应用; 2、熟练掌握二叉排序树的特性、构造方法以及查找、插入和建树运算的算法实现; 3、理解二叉排序树的删除运算以及算法的实现; 4、理解平衡二叉树的有关概念以及建树的方法; 5、熟悉各种查找方法,学会根据实际问题的要求,选取合适的查找方法及相应的存储结构来解决具体问题,能够完成主要查找方法的编程任务。	回顾本节课的知识点,构建知识架构。	3min	**设计思路:**帮助学生构建一个完整的知识体系。解发学生更多更深的思考,完成拓展练习。
项目分析				

14	课堂连线企业导师，从实际应用范围的角度对几种查找算法的优劣性进行对比评价。分析学生成绩管理系统的进一步优化策略和方向，点明二叉排序树在系统中的作用，说明评价标准，强调注意事项。	根据企业导师点评进行自我反思，及时进行自我学习调整，分析本节课项目实现的设计要点，进行算法设计。	8min	**教学设计思路：** 企业导师根据行业标准进行项目分析及点评，帮助学生及时发现问题、解决问题，强化对知识的理解，提升学生的职业素养。

八、知识拓展

1、拓展训练：

在学生成绩管理系统中，假设电子通讯录包含学生姓名、学院、学号、电话等信息，采用记事本存储具体信息，内容图所示，在大量的信息中我们如何能够根据要求快速查找得到想要的结果呢？分别可以显示全部通讯录信息、通过姓名查找通讯录信息、通过学院查找通讯录信息，也可以查找学生学号<20 的通讯录信息。

2、考证（考研）训练：

（2009 年考研真题）下列叙述中，不符合 m 阶 B 树定义要求的是（ ）。

A 根结点最多有 m 棵子树

B 所有叶子结点都在同一层上

C 各结点内关键字均升序或降序排列

D 叶子结点之间通过指针链接

（软件水平考证）在平衡二叉树中插入一个结点就造成了不平衡，设最低的不平衡结点为并已知 A 的左孩子的平衡因子为一 1，右孩子的平衡因子为 0,则为了使其平衡，应做（ ）。

A LL 型调整 B RR 型调整结构 C RL 型调整 D LR 型调整

3、竞赛训练：

蓝桥杯查找部分 https://dasai.lanqiao.cn/notices/1096/

九、教学评价与反馈

本节课的教学过程中，虽然用到一些教学手段，起到一定的作用，但是本节课的内容本身就难，需要学生较强的逻辑思维，需要学生熟练掌握编程语言，因此教师艰难的教，学生吃力的学。要课后练习中，分多次布置作业，将学习目标细化，首先，让学生理解查找的逻辑结构及特点，以及各种操作的算法思想；其次，让学生动用编程语言编写程序实现这些算法；最后，把这些编程思想和设计思路运用到实际应用中斛问题。

十、教学总结

本课程主要讲授了树表查询的概念、算法和应用，包括二叉排序树、平衡二叉树和 B 树等数据结构。学生在学习中需要理解树表查询的基本原理和特点，掌握树表查询的实现方式和算法，了解树表查询在实际应用中的场景和优缺点，并能够根据需求选择合适的数据结构进行操作。

本课程的教学重点包括理解树表查询的概念和基本原理，掌握树表查询的实现方式和算法，了解树表查询在实际应用中的场景和优缺点。在教学中，需要通过具体实例和图示来讲解和演示，帮助学生理解和掌握树表查询的概念和原理；通过实践和训练来提高学生的编程能力和代码调试能力，帮助学生掌握树表查询的实现方式和算法；通过案例分析和讨论来深入了解树表查询在实际应用中的场景和优缺点，提高学生的应用能力和创新能力。

教学难点主要在于初学者可能对树表查询的概念和原理理解和掌握困难，实现和调试树表查询算法需要具备较高的编程能力和代码调试能力，了解树表查询在实际应用中的场景和优缺点需要具备一定的实际经验和知识储备。因此，教师需要采用多种教学手段和策略，如图示、实践、案例分析和讨论等，帮助学生理解和掌握树表查询的概念和原理，提高学生的编程能力和代码调试能力，增强学生的应用能力和创新能力，从而达到教学目标。

十一、项目实现代码

```
//函 数 名: search_course
//功    能: 按指定课程名称搜索拥有该科目的学生，并将其信息复制到新二叉树
//参    数:     student: 搜索的学生
//              subjust: 该学生的科目二叉树
//            temp_tree: 新的二叉树
//          course_name: 指定的课程名称
void search_course(STUDENT *student, COURSE *subjust, STUDENT *temp_tree, char course_name[15])
{
   if (subjust == NULL)     //若为 NULL 则返回
   {      return;    }
   search_course(student, subjust->lnext, temp_tree, course_name);     //往左走继续递归
   search_course(student, subjust->rnext, temp_tree, course_name);     //往右走继续递归
   if (strcmp(subjust->course, course_name) == 0)     //比较两个课程的名称，若相等则找到了相对应课程
   {
      if (temp_tree->student_number == 0)    //若学号为 0，表示为 temp_tree 空树
      {  memcpy(temp_tree, student, sizeof(STUDENT));     //将该学生的信息复制到空树，作为根
         temp_tree->lnext = NULL;     //左右指针空指
```

```
                temp_tree->rnext = NULL;
                COURSE *temp_subjust = (COURSE *)malloc(sizeof(COURSE));          //申
请科目结构体
                 memcpy(temp_subjust, student->subjust, sizeof(COURSE));      //复制学
生二叉树根信息到新科目结构体
                temp_subjust->lnext = NULL;      //新科目结构体左右指针空指
                temp_subjust->rnext = NULL;
                copy_course(student->subjust->lnext, temp_subjust, 1);      //调用函数
将该学生的科目二叉树左边复制到新科目结构体
                copy_course(student->subjust->rnext, temp_subjust, 2);      //调用函数
将该学生的科目二叉树右边复制到新科目结构体
                temp_tree->subjust = temp_subjust; //学生根的科目指针指向新科目结构体
            }
            else
            {STUDENT *temp_student = (STUDENT *)malloc(sizeof(STUDENT));      //申请
显示结构体
                memcpy(temp_student, student, sizeof(STUDENT));      //将该学生的信息
复制到新学生结构体
                temp_student->lnext = NULL;      //左右指针空指
                temp_student->rnext = NULL;
                COURSE *temp_subjust = (COURSE *)malloc(sizeof(COURSE));      //申请科
目结构体
                memcpy(temp_subjust, student->subjust, sizeof(COURSE));      //复制该
学生的信息到新科目结构体
                temp_subjust->lnext = NULL;      //左右指针空指
                temp_subjust->rnext = NULL;
                copy_course(student->subjust->lnext, temp_subjust, 1);      //调用函数
将该学生的科目二叉树左边复制到新科目结构体
                copy_course(student->subjust->rnext, temp_subjust, 2);      //调用函数
将该学生的科目二叉树右边复制到新科目结构体
                temp_student->subjust = temp_subjust;      //学生根的科目指针指向新科
目结构体
                temp_tree = insert_temp_student_by_subjust(temp_student, temp_tree,
subjust->score, course_name);      //调用函数将新学生结构体插入二叉树
            }   }
    }
```

《数据结构及算法设计》教案 8，17-18 学时

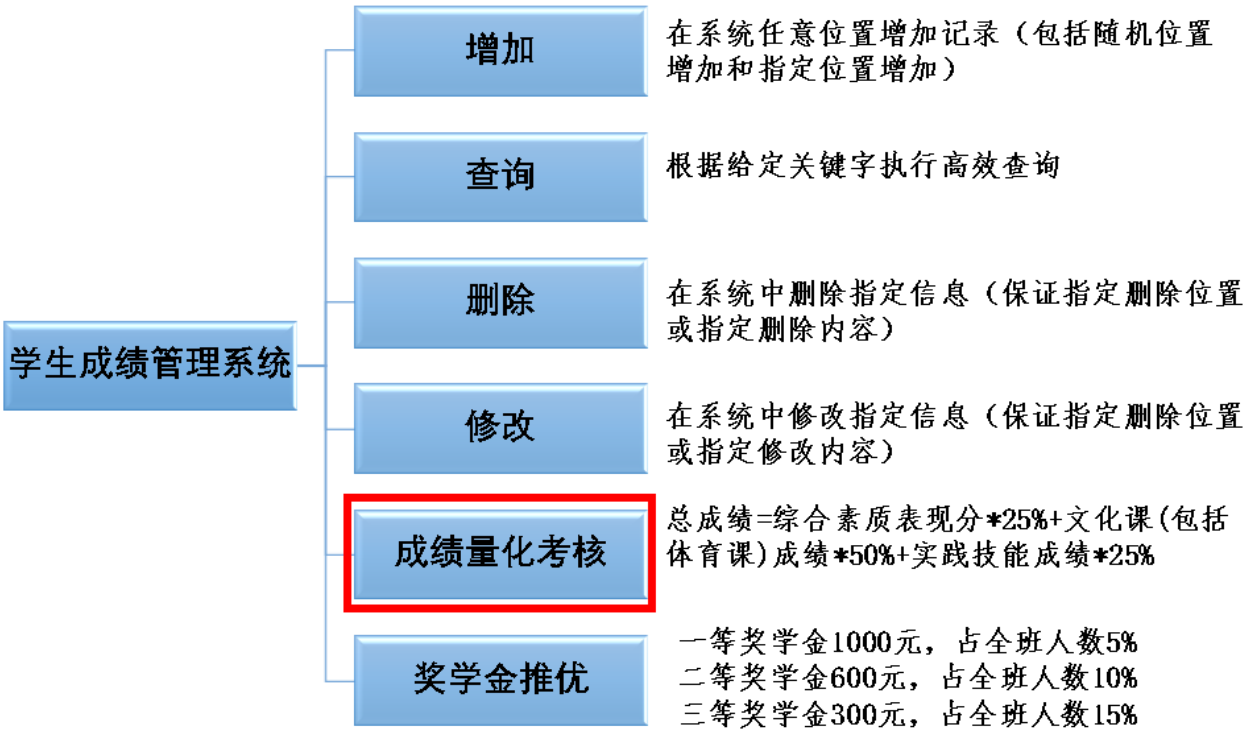
<table>
<tr><th colspan="4">一、教学基本信息</th></tr>
<tr><td>课程名称</td><td>二叉树的应用</td><td>授课教师</td><td>申妙芳</td></tr>
<tr><td>授课班级</td><td>21 级计算机应用工程 3 班</td><td>授课时数</td><td>2</td></tr>
<tr><td>授课时间</td><td>周二 3-4 节</td><td>授课地点</td><td>81107</td></tr>
<tr><th colspan="4">二、教学分析</th></tr>
<tr><td>教学内容</td><td colspan="3">1、哈夫曼树相关概念；
2、带权路径长度的计算；
3、构造哈夫曼树；
4、哈夫曼树的算法实现；
5、哈夫曼编码的方法。</td></tr>
<tr><td>学情分析</td><td colspan="3">本节课是树的第四部分内容，也是树的最后一个知识点。在此之前，学生已经对树及二叉树的概念、存储结构、行为特点以及遍历方式有了较为全面的了解。
树的应用及其广泛，除了可以利用其结构特点实现族谱等层级关系的表示、优化学生成绩管理系统中数据查找功能提升系统运行效率外，在通讯领域也可以实现高效的编码译码。因此，本节课在学生已构建的树和二叉树的知识架构上进行扩充，让学生对树的应用范围有更清晰的了解和认知。
由于树的数据结构比线性结构更为复杂和抽象，因此学生在学习这部分内容时，部分同学难以建立空间想象，在教学过程中需要频繁与学生互动，防止学生走神，对核心内容用动画形象展示出来。</td></tr>
<tr><th colspan="4">三、教学目标确定</th></tr>
<tr><td rowspan="3">教学目标</td><td>知识目标</td><td colspan="2">1、掌握二叉树带权路径长度的计算方法、哈夫曼算法思想；
2、熟悉哈夫曼编码；
3、了解哈夫曼算法在文件及图像压缩等方面的应用；
4、掌握哈夫曼树算法的实现。</td></tr>
<tr><td>能力目标</td><td colspan="2">1、数据结构与算法的灵活应用能力，用计算机解决实际问题的能力；
2、持续学习、独立思考、团队思考的基本能力；
3、锻炼学生动手能力，从而获得对计算机技术的认同感。</td></tr>
<tr><td>素质目标</td><td colspan="2">1、培养学生具备辩证思维的能力；
2、培养学生具有热爱科学、勇于实践、实事求是的学风和创新意识、创新</td></tr>
</table>

		精神； 3、培养学生具有较强的执行能力以及较高的工作效率和安全意识； 4、培养学生规范、严谨、精确的工作态度和情感。
教学重点		哈夫曼树及其应用，构造哈夫曼树、哈夫曼编码的方法及带权路径长度的计算。
教学难点		哈夫曼树及其应用，构造哈夫曼树、哈夫曼编码的方法及带权路径长度的计算。

四、思政融入课堂

1、通过讲述哈夫曼算法提出者——戴维•哈夫曼的科学研究故事，以此开展理想信念教育和人生价值教育，诠释“奋斗的青春最美丽”，引导学生树立积极向上的人生观和世界观；
2、讲述哈夫曼算法在成绩等级判定程序及电文编码的应用实例，让学生懂得面对问题时，选择最合理的算法对问题进行求解，提升知识运用能力。

五、课程在项目中的定位

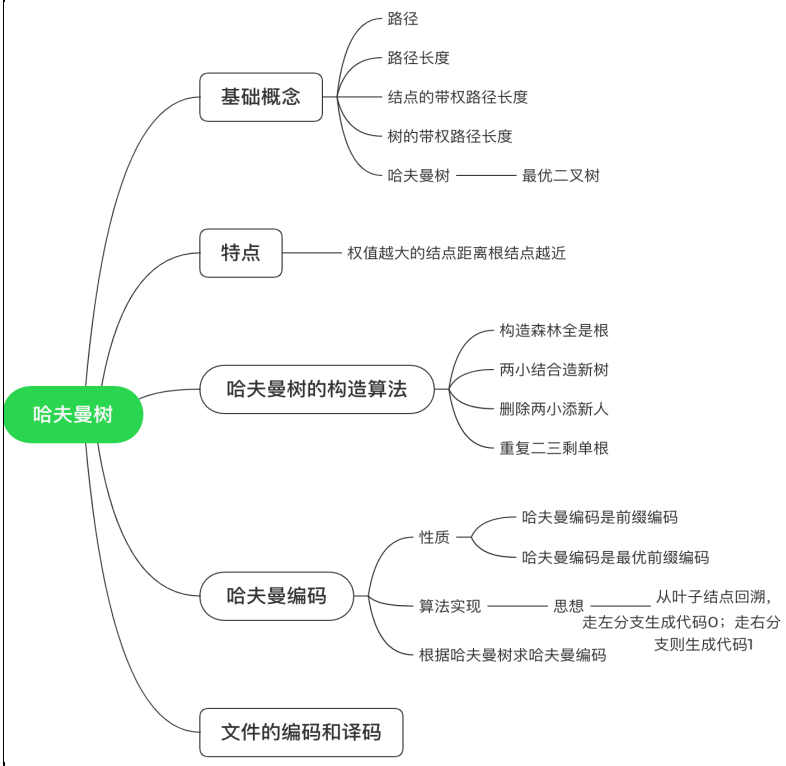


利用哈夫曼树对成绩考核等级判定算法进行优化，提高等级判定效率，从而提升查询效率，综合优化系统整体功能。

六、教学策略

设计思路

1、思维导图：



2、具体思路：

（1）通过案例分析，引导学生思考不同形态的二叉树对程序运行效率的影响；

（2）对哈夫曼树（最优二叉树）进行定义，了解其判定原则；

（3）利用动画演示哈夫曼树的构造过程。将构造过程以顺口溜的形式进行讲解，方便学生记忆；

（4）根据哈夫曼树构造过程画出程序流程图，通过知识迁移设计算法并上机实践，分析算法复杂度；

（5）拓展哈夫曼树的应用实例，构造哈夫曼编码，实现高效的编码译码功能；

（6）头脑风暴，讨论哈夫曼树在图像处理、信号处理、计算机网络等领域的应用，以及哈夫曼树的实现原理及其在不同应用中的优势。

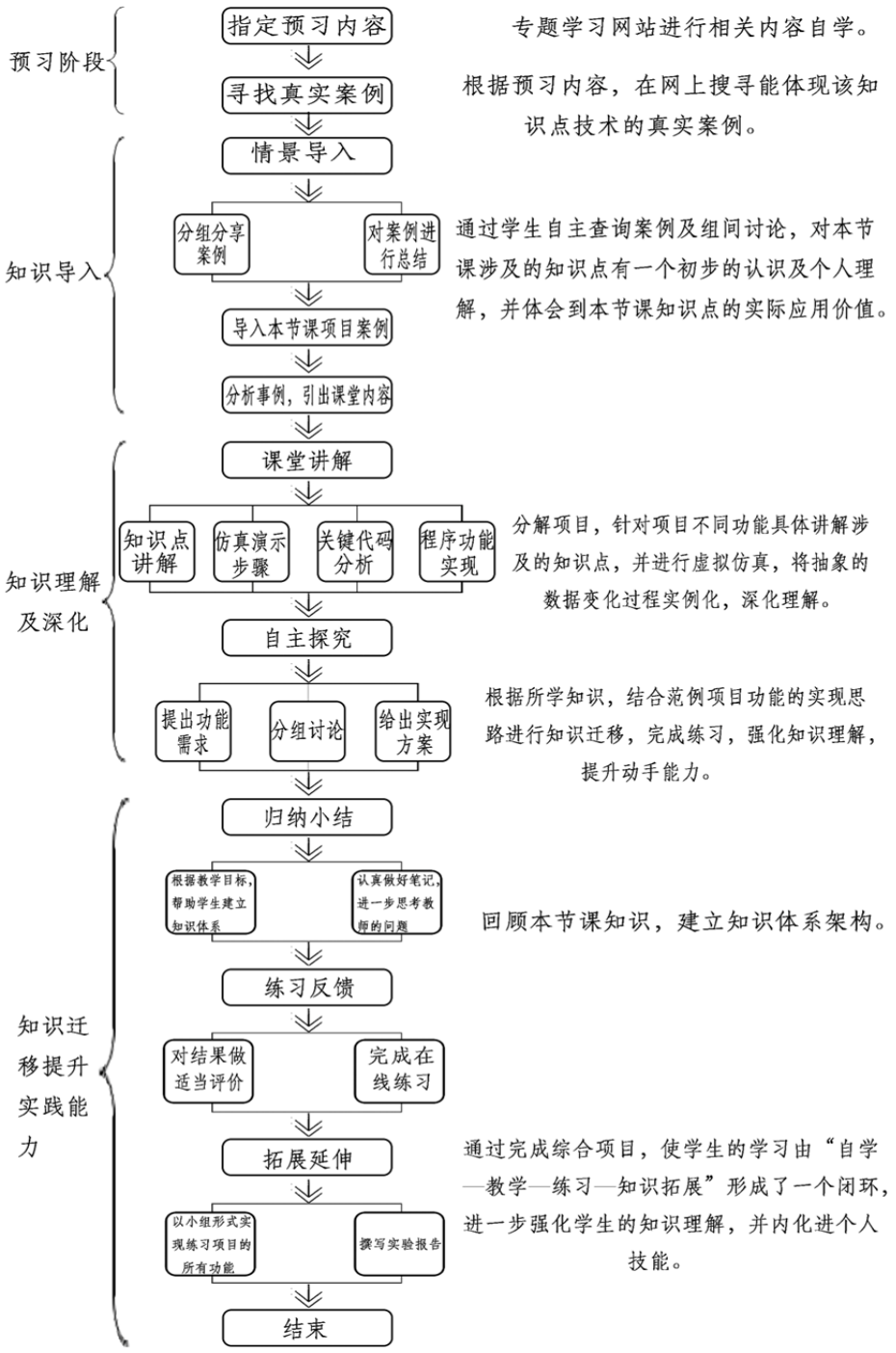
3、教学方法：

课堂讲授以任务导入、学生分析为主要手段。教师在学生讨论过程中适时加以引导和总结，结合课堂练习、头脑风暴等辅助教学方式，增加课堂信息量，加深学生对编码方法、性质的理解，利用多媒体课件的文字、动画由浅入深地演示知识点，并同步地讲授知识点的内涵；对于教学难点，注重图文并茂的方式进行深入剖析，并辅以具体事例加以验证说明；对于教学重点，如哈夫曼编码过程，进行板书，并安排课堂和课后练习。

教学资源

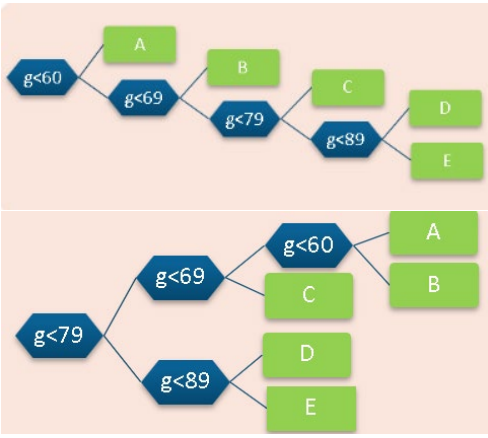
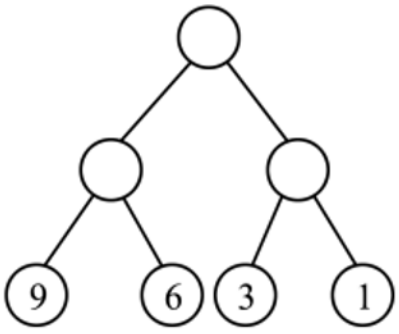
学习环境：机房、局域网，交互式电子黑白

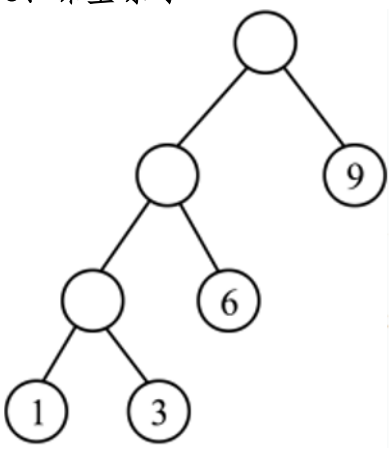
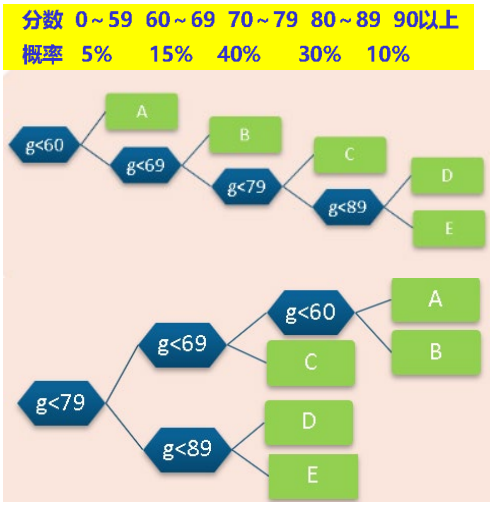
学习资源：

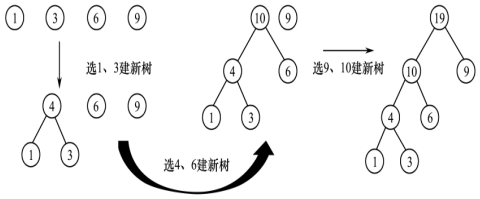
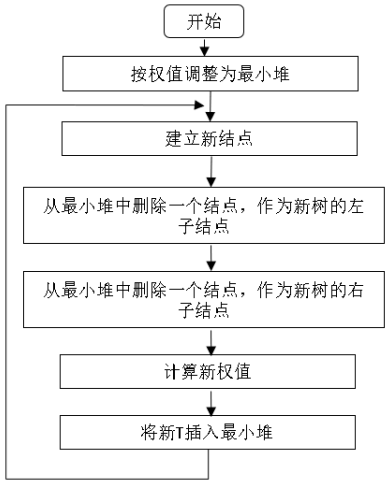
	1、专题学习网站：包括慕课网、学习通、职教云等； 2、授课课件：根据此节学习内容制作的 PPT 课件； 3、多媒体资源库：虚拟仿真演示案例、项目演示案例； 4、案例库：课堂练习题库、测验题库等； 5、精品课程网站：本门课程的精品课程网站； 6、VC6.0++运行环境：进行案例演示及学生练习项目的专业运行环境。
教学流程	

七、教学过程

<table>
<tr><th rowspan="2">教学环节</th><th colspan="2">教学活动</th><th rowspan="2">时间安排</th><th rowspan="2">设计思路与教学手段</th></tr>
<tr><th>教师授课内容</th><th>学生</th></tr>
<tr><td colspan="5">课前预习</td></tr>
<tr><td>1</td><td>1、提供慕课网资源：请同学们预习本次课程基本知识，如哈夫曼的定义，带权路径长度的计算等。
2、思政学习：通信系统中的信道编码技术
https://www.elecfans.com
/d/1835144.html
3、提出问题：哈夫曼树的应用场景有哪些。</td><td>1、在充足的时间内观看慕课视频；
2、完成精品课成基础练习；
3、分组收集哈夫曼树的应用案例。</td><td>课前一周</td><td>设计思路：利用慕课网自主学习，提高学生学习主动性和创造性，强化解决问题的能力。
涉及教学资源：
慕课网、搜索引擎。</td></tr>
<tr><td colspan="5">知识导入</td></tr>
<tr><td>2</td><td>分析学生案例，探讨哈夫曼树在现实生活中的应用。</td><td>以组为单位分享哈夫曼树的实际应用案例。</td><td>3min</td><td>教学手段：
分组讨论、真实案例。</td></tr>
<tr><td>3</td><td>引出课堂案例：
将百分制的考试成绩（grade）转换成五分制的成绩，转换规则如下表：
<table><tr><td>分数grade</td><td>grade<60</td><td>60<=grade<69</td></tr><tr><td>五分制level</td><td>不及格</td><td>及格</td></tr></table><table><tr><td>70<=grade<79</td><td>80<=grade<89</td><td>grade>90</td></tr><tr><td>中</td><td>良</td><td>优</td></tr></table>思考，如何设计算法可以提高程序运行效率。</td><td>观察案例，绘制成绩判定流程图，思考如何用C语言实现，查询效率又如何。</td><td>2min</td><td>设计思路：成绩转换是计算机程序设计中非常经典的案例，学生在此之前已经非常熟悉其程序实现原理。选用此案例，引起学生好奇心，尝试用新知识解决老问题，主动探索解决方案。</td></tr>
<tr><td>4</td><td>分析案例，引出新课：
分数 0~59 60~69 70~79 80~89 90以上
概率 5% 15% 40% 30% 10%
考虑实际成绩的分布情况，教师提供</td><td>观察案例，组内讨论哪种结构形式的判断效率</td><td>2min</td><td>设计思路：现阶段，学生对于程序的设计往往脱离现实。因此，当经典题目与现</td></tr>
</table>

	两种不同的结构方式，供学生对比其程序运行效率。  	更高。		实相结合时，往往会颠覆学生的固定认知，从而站在一个更高的角度看待问题，极大的引起学生的求知欲。同时，也培养了学生多方位思考问题的习惯。
知识讲解				
5	**板书重要知识** **哈夫曼树相关基本概念** **1、路径：**从树中一个结点到另一个结点之间的分支序列构成两个结点间的路径； **2、路径长度：**路径上分支的条数称为路径长度； **3、树的路径长度：**从树根到每个结点的路径长度之和称为树的路径长度； **4、结点的权：**给树中结点赋予一个数值，该数值称为结点的权； **5、带权路径长度：**结点到树根间的路径长度与结点的权的乘积，称为该结点的带权路径长度； **6、树的带权路径长度：**树中所有叶子结点的带权路径长度之和，称为树的带权路径长度，通常记为 WPL：$WPL=\sum_{k=1}^{n} W_k \times L_k$ 其中，n 为叶子数，W_k 为第 k 个叶子的权值，L_k 为第 k 个叶子到树根的路径长度； **7、举例：**  	1、在回答教师问题过程中复习路径、路径长度、树的路径长度、权等相关概念； 2、利用已学知识计算带权二叉树的 WPL，巩固理论知识。	10min	**设计思路：** 1、问答与练习 本小节内容主要为基本概念讲解。由于学生之前已经预习过相关知识，所以主要以问答的形式进行知识复习，确保学生的课堂参与度。现学现练的形式可以很好的帮助学生巩固所学知识，培养学生的主观能动性和创新能力； 2、板书 给学生“留白”，利用板书时间思考、消化教学内容。 **教学资源：** PPT 讲解，案例分析，虚拟仿真互动。

	WPL=9x2+6x2+3x2+1x2=38; **8、课堂练习**  请问这棵二叉树的 WPL 是多少？			
6	**1、案例分析：**  利用 WPL 知识，判断哪种结构的程序运行效率更高？ **2、知识拓展**  **3、头脑风暴** 能否继续改变判定树的结构，使其程序运行效率最高？	1、计算两种判定树的 WPL，找出运行效率较高的判定树； 2、根据问题，计算 WPL 值，寻找带权路径最小的二叉树； 3、头脑风暴，尝试继续改变判定树的结构，在结果不变的前提下，利用 WPL 知识，寻找运行效率最高的判定树。	5min	**设计思路：** 1、案例分析 将 WPL 知识应用于案例分析，既巩固了所学知识，也有效的支撑了学生在之前对两种结构的讨论结果，增加自我效能感； 2、头脑风暴 在对判定树结构进一步改造的过程中，小组间的知识碰撞能够极大激发学生的探索热情，提升课堂参与度。 **教学资源：** PPT 讲解，案例分析。
7	**1、哈夫曼树的定义：** 给定 n 个权值作为 n 个叶子结点，构造一棵二叉树，若带权路径长度达到最小，称这样的二叉树为最优二叉树，也称为哈夫曼树。 **2、哈夫曼树的建立：**	1、了解哈夫曼树的定义； 2、观看哈夫曼树构建动画，小组	25min	**设计思路：** 1、动画演示 哈夫曼树的构建涉及到树的重复添加和删除，是较为抽象的一个知识点。通过对结

	（1）动画展示哈夫曼树的构建过程； （2）课堂讨论总结构建步骤； （3）教师通过图示法总结构建步骤，给出构建口诀。 构造森林全是根，两小结合造新树，删除两小添新人，重复二三剩单根。  **3、课堂练习** 职教云平台习题，假定 4 个带权叶子结点的权值依次为 2,4,6,9.按此生成一棵哈夫曼树,此路径长度 WPL 为(　)。 **4、哈夫曼算法的实现** （1）学生口述创建过程，根据创建步骤构建程序流程图：  （2）采用相应的存储形式，对结点进行定义： #define n 30 #define M 2*n-1 Typedef struct{ Int weight; Int parent; Int Lchild; Int Rchild; }HTNode, HuffmanTree[M+1] （3）分析关键代码段。	讨论总结构建步骤； 3、跟随老师一起复习哈夫曼树构建构成，深化理解； 4、独立完成课堂练习； 5、回忆并口述构建步骤，组内讨论程序实现过程； 6、利用已学树的知识，根据程序流程图分析并编写关键代码段。		点动态变化过程的演示，帮助学生建立抽象的空间想象，直观感受到从单独的若干结点到形成一棵树的动态变换过程； 2、分组讨论，帮助学生在讨论总结的过程中内化知识点，建立自己的知识架构； 3、图例演示。教师通过图例对构建步骤进行演示，规范步骤，强化学生对知识的理解； 4、课堂练习，帮助学生内化知识点。 教学资源： PPT 讲解， 示例图演示，虚拟仿真，编程环境
8	**项目实现：** 分数 0~59 60~69 70~79 80~89 90以上 概率 5% 15% 40% 30% 10%	1、结合程序流程图，	20min	**教学设计思路：** 1、知识迁移，程序

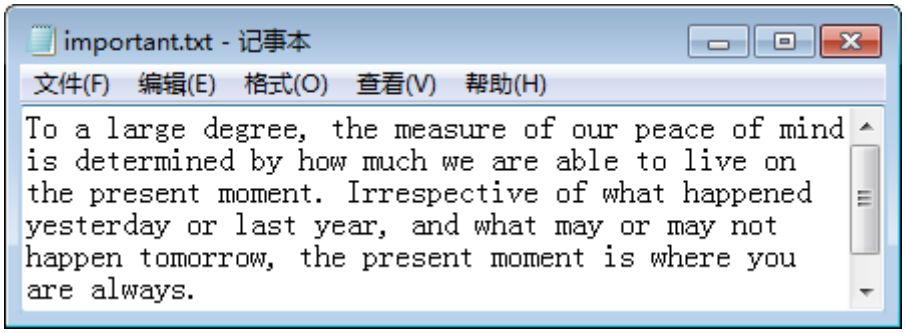
	5个分数段代表5个独立结点，分数段的概率为该结点的权值，请构建哈夫曼树，提高分数查询程序的运行效率。 **教师巡堂，个性化辅导。**	理解程序段的含义； 2、项目实践，小组合作，建立哈夫曼树，实现高效的成绩转换功能。		的实现帮助学生在逻辑结构和功能实现之间建立连接，将理论知识转移为实践能力； 2、培养学生的团队合作精神和团队意识。 **教学资源：** PPT讲解， 示例图演示，上机编程并运行。
9	**知识拓展** **哈夫曼编码** **1、案例导入：** 对字符串“ABBACAACBD”进行编码。对于字符集{A，B，C，D}，若采用{1，01，000，001}进行编码，请问字符串编码后的长度是多少？ 分析该字符串，考虑到字符使用频率的不同，根据其使用频率高低对字符集{A，B，C，D}采用{0，1，01，11}进行编码，请问字符串编码后的长度是多少？ 请问采用第二种形式编码后，能否译码回原始文本串？ **2、案例分析**，引出哈夫曼编码。	1、分组讨论两种编码的特点，以及编码译码的正确率； 2、头脑风暴，讨论哈夫曼树在编码中的作用。	5min	**教学设计思路：** 1、案例分析 以简单的案例为切入点引发学生的思考，认识的不同的编码对通讯效率和译码正确率的影响。理论知识与生活实际相结合，培养学生全面思考问题的能力，拓展思维； 2、头脑风暴 在讨论中完成知识的碰撞，深化对知识的理解。 **教学资源：** PPT讲解。
10	**哈夫曼编码的原理** **1、构造哈夫曼树；** **2、哈夫曼树编码规定：**在哈夫曼树上求各叶子结点的编码，在哈夫曼树中约定左分支表示符号‘0’，右分支表示符号‘1’，用根结点到叶子结点路径上的分支符号组成的串，作为叶子结点字符的编码，这就是哈夫曼编码。 **3、案例分析与实践：** 分析字符串“ABBACAACBD”，以每个字符出现的次数作为结点权值，对该字符串进行编码。 **4、知识拓展：** 一棵哈夫曼树共有 29 个结点,对它进	1、理解哈夫曼编码； 2、知识迁移，完成项目实践； 3、根据所学知识完成知识拓展； 4、发散性思维，讨论哈夫曼树的其他应用。	8min	**教学设计思路：** 1、知识迁移及拓展； 将哈夫曼树知识迁移至哈夫曼编码的实现。在巩固知识的同时拓展学生思维，锻炼学生思维灵活性； 2、头脑风暴 根据哈夫曼树的特点思考其应用范围，锻炼学生发散性思维，以及对知识的灵活应用的能力。

	行哈夫曼编码,能够得到( )种不同的编码。 **5、课堂讨论：**哈夫曼树的应用还有哪些？ 哈夫曼树广泛应用于传真机,图像压缩和计算机安全等领域。			**教学资源：** PPT 讲解，案例分析。
课堂小结				
11	哈夫曼树又称最优二叉树，是带权路径长度最短的树，可用来构造最优编码，用于信息传输、数据压缩等方面，是一种该用广泛的二叉树。掌握哈夫曼树的构造，理解构造哈夫曼树的算法。	回顾本节课的知识点，构建知识架构。	3min	**设计思路：**帮助学生构建一个完整的知识体系。解发学生更多更深的思考，完成拓展练习。
项目分析				
12	课堂连线企业导师，从实际应用范围的角度对几种查找算法的优劣性进行对比评价。分析学生成绩管理系统的进一步优化策略和方向，点明二叉排序树在系统中的作用，说明评价标准，强调注意事项。	根据企业导师点评进行自我反思，及时进行自我学习调整，分析本节课项目实现的设计要点，进行算法设计。	8min	**教学设计思路：** 企业导师根据行业标准进行项目分析及点评，帮助学生及时发现问题、解决问题，强化对知识的理解，提升学生的职业素养。

八、知识拓展

1、拓展训练：

日常工作中经常有一些重要的文件，为了保证安全性通常需要对其进行加密处理，使用时再进行解密处理，例如，有一个文本文件 important.txt，如图所示，内容由英文字母、英文逗号、英文句点、空格和回车组成，要求加密的文件进行解密后与原文件要完全一致。



2、考证（考研）训练：

（2019 年考研真题）设有 13 个值,用它们组成一棵哈夫曼树 ，则该哈夫曼树共有_____个结点。

A) 13 B) 12 C) 26 D)25

（软件水平考证）

由权值分别为 11，8，6，2，5 的叶子结点生成一棵哈夫曼树，它的最小带权路径长度为（ ）。

A) 24 B) 71 C) 48 D)63

3、职业训练：

在学生管理系统登录模块的设计中，需要将学生输入的密码进行加密，加密后保存到数据库中，假设学生传送的密码由 a,b,c,d,e,f,g,h,i9 个字母组成，字母按传输中出现的频率分别为：{5,6,7,8,9,10,15,18,22}。

（1）请画出哈夫曼树；

（2）请为这九个字母设计哈夫曼编码。

4、竞赛训练：

蓝桥杯哈夫曼树部分 https://dasai.lanqiao.cn/notices/1096/

九、教学评价与反馈

学生自主评价：

1、通过完成课前预习及课后习题测试、知识拓展、讨论分享学习心得及算法实训之后展开评价；

2、是否通过课外自主学习明确了本节课的教学目标；

3、是否掌握了哈夫曼算法的基本思想及哈夫曼编码；

4、是否通过实例应用了解了哈夫曼算法应用场合；

5、是否体会到了科学家研究探索精神。

教学效果评价：

1、课前布置学生在问卷星上回答问题，通过答题情况了解学生是否复习相关知识以及是否了解本节课相关内容；

2、课堂中观察学生的注意力、参与度与投入度，教学互动环节中观察学生的反应情况及 MOOC 平时的随堂练习情况；

3、课后通过问卷星平台查看学生作业情况，通过算法在线实训平台了解学生实验情况，通过中国大学 MOOC 网平台了解学生对知识的学习情况，通过班级群了解学生提问情况。

十、教学总结

哈夫曼树（Huffman Tree）是一种常用的用于实现编码的树形结构，它是一种最优二叉树，也称作最优二叉树或最佳二叉树。

哈夫曼树的构造是一种贪心算法，它的基本思想是：将权值最小的两个结点合并成一个新的结点，并将新结点的权值设置为其子结点的权值之和，然后将新结点放入原来的结点集合中，重复上述操作，直到只剩下一个结点为止。

哈夫曼树的应用非常广泛，它可以用于文件压缩、编码、数据传输等。它的优势在于可以有效地减少数据传输的时间和空间，从而提高数据传输的效率。

由于本节内容本身较难，大多数同学理解吃力，针对这一现象，继续布置相关学习视频资料，多点编程任务，加大上机练习时间，逐步理解消化本节课内容。

十一、项目实现核心代码

```
(算法 6.25 建立哈夫曼树)
void CrtHuffmanTree(HuffmanTree ht , int w, int n)
{ m=2*n-1;
  for(i=1;i<=n;i++) ht[i]={w[i],0,0,0};     /*初始化前 n 个元素成为根结点*/
  for(i=n+1;1<=m;i++) ht[i]={0,0,0,0};      /*初始化后 n-1 个空元素*/
  for(i=n+1;i<=m;i++)                       /*从第 n+1 个元素开始构造新结点*/
  { select(ht,i-1,&s1, &s2);
      /*在 ht 的前 i-1 项中选双亲为 0 且权值最小的两结点 s1、s2*/
    ht[i].weight=ht[s1].weight+ht[s2].weight;   /*建新结点,赋权值*/
    ht[i].Lchild=s1;ht[i].Rchild=s2;            /*赋新结点左、右孩子指针*/
    ht[s1].parent=1; ht[s2].parent=i;           /*改 s1、s2 的双亲指针*/
  }
}
(算法 6.26 哈夫曼编码)
void CrtHuffmanCodel(HuffmanTree ht , HuffmanCode hc, int n)
/*从叶子到根,逆向求各叶子结点的编码*/
{ char *cd;
  int start;
  cd=(char *)malloc(n*sizeof(char));            /*临时编码数组*/
  cd[n-1]='\0';                                 /*从后向前逐位求编码,首先放编码结束符*/
  for(i=1;i<=n;i++)                             /*从每个叶子开始,求相应的哈夫曼编码*/
  { start=n-1;c=i;p=ht[i].parent;               /*c 为当前结点,p 为其双亲*/
    while(p! =0)
    { --start;
    if(ht[p].LChild==c) cd[start]='0';          /*左分支得'0'*/
    else cd[start]='1';                         /*右分支得'1'*/
    c=p;p=ht[p].parent;                         /*上溯一层 */
    }
    hc[i]=(char *)malloc((n-start)*sizeof(char)); /*动态申请编码串空间*/
    strcpy(hc[i], &cd[start]);                  /*复制编码*/
  }
  free(cd);
}
```