河北地质大学

实验指导书

2020/2021 学年　　第一学期

课程名称：数据结构

授课对象：软件工程专业 二年级

编制教师：吴 聪 聪

授课学时：16

数据结构实验指导书

一、实验目的

《数据结构 》是计算机专业一门重要的专业技术基础课程，是计算机专业的一门核心的关键性课程。本课程较系统地介绍了软件设计中常用的数据结构以及相应的存储结构和实现算法，介绍了常用的多种查找和排序技术，并做了性能分析和比较，内容非常丰富。本课程的学习将为后续课程的学习以及软件设计水平的提高打下良好的基础。

由于以下原因，使得掌握这门课程具有较大的难度:

（ 1 ）内容丰富，学习量大，给学习带来困难;

（ 2 ）所用到的技术多，而在此之前的各门课程中所介绍的专业性知识又不多，因而加大了学习难度;

（ 3 ） 隐含在各部分的技术和方法丰富，也是学习的重点和难点。

根据《数据结构》课程本身的技术特性，设置《数据结构课程实验》实践环节十分重要。通过实验实践内容的训练，突出构造性思维训练的特征 ， 目的是提高学生组织数据及编写大型程序的能力。

课程上机实验的目的，不仅仅是验证教材和讲课的内容， 检查自己所编的程序是否正确， 课程安排的上机实验的目的可以概括为如下几个方面:

1、加深对课堂讲授内容的理解

实验是对学生的一种全面综合训练。是与课堂听讲、自学和练习相辅相成的必不可少的一个教学环节。通常，实验题中的问题比平时的习题复杂得多，也更接近实际。实验着眼于原理与应用的结合点，使学生学会如何把书上学到的知识用于解决实际问题，培养软件工作所需要的动手能力；另一方面，能使书上的知识变"活"，起到深化理解和灵活掌握教学内容的目的。

不少学生在解答习题尤其是算法设计题时，觉得无从下手，做起来特别费劲。实验中的内容和教科书的内容是密切相关的，解决题目要求所需的各种技术大多可从教科书中找到，只不过其出现的形式呈多样化，因此需要仔细体会，在反复实践的过程中才能掌握。

2、培养学生软件设计的综合能力

通过实验使学生不仅能够深化理解教学内容，进一步提高灵活运用数据结构、算法和程序设计技术的能力，而且可以在需求分析、总体结构设计、算法设计、程序设计、上机操作及程序调试等基本技能方面受到综合训练。实验着眼于原理与应用的结合点，使学生学会如何把书本上和课堂上学到的知识用于解决实际问题，从而培养计算机软件工作所需要的动手能力。

3、熟悉程序开发环境，学习上机调试程序

一个程序从编辑，编译，连接到运行，都要在一定的外部操作环境下才能进行。所谓"环境"就是所用的计算机系统硬件，软件条件，只有学会使用这些环境， 才能进行程序开发工作。通过上机实验，熟练地掌握程序的开发环境，为以后真正编写计算机程序解决实际问题打下基础。同时，在今后遇到其它开发环境时就会触类旁通，很快掌握新系统的使用。

完成程序的编写，决不意味着万事大吉。你认为万无一失的程序，实际上机运行时可能不断出现麻烦。如编译程序检测出一大堆语法错误。有时程序本身不存在语法错误，也能够顺利运行，但是运行结果显然是错误的。开发环境所提供的编译系统无法发现这种程序逻辑错误，只能靠自己的上机经验分析判断错误所在。程序的调试是一个技巧性很强的工作，尽快掌握程序调试方法是非常重要的。分析问题，选择算法，编好程序，只能说完成一半工作，另一半工作就是调试程序，运行程序并得到正确结果。

二、实验要求

常用的软件开发方法，是将软件开发过程划分为分析、设计、实现和维护四个阶段。虽然数据结构课程中的实验题目的远不如从实际问题中的复杂程度度高，但为了培养一个软件工作者所应具备的科学工作的方法和作风，也应遵循以下五个步骤来完成实验题目：

1．问题分析

在进行设计之前，首先应该充分地分析和理解问题，明确问题要求做什么?限制条件是什么。本步骤强调的是做什么?而不是怎么做。对问题的描述应避开算法和所涉及的数据类型，而是对所需完成的任务作出明确的回答。

2．逻辑设计和详细设计

逻辑设计指的是，对问题描述中涉及的操作对象定义相应的数据类型，并按照以数据结构为中心的原则划分模块，定义主程序模块和各抽象数据类型；详细设计则为定义相应的存储结构并写出各函数的伪码算法。

3．编码实现和静态检查

做完详细设计后，根据设计的存储结构和算法，编程代码，完成后检查代码的逻辑是否有错。在实验环境中编译检查是否存在语法错误。

4．调试和测试

调试程序使程序正常运行，用测试用例测试程序的正确性，如果有问题，再反复检查程序的逻辑是否有问题，必要时要单步调试。

5．总结和整理实验报告

实验结束后，要整理实验结果并认真分析和总结，根据教师要求写出实验报告。实验报告格式见附录 A。实验评分标准见附录 B。

实验安排:

序号	实验项目	实验学时	实验类型
1	线性表	2	设计性实验
2	栈和队列	2	设计性实验
3	树与二叉树	4	验证性实验
4	图	4	设计性实验
5	查找与排序	4	设计性实验

实验一　线性表及其的基本操作

实验学时：2 学时

一、实验目的和要求

1、熟练掌握线性表的基本操作在顺序存储和链式存储上的实现。

2、掌握顺序表的定义以及各种基本操作（初始化、插入、删除、定位，查找等）的实现。

3、掌握单链表的定义以及各种基本操作（初始化、插入、删除、定位，查找等）的实现。

二、实验内容（任选一题完成即可）

1). 用线性表表示整数集合 A, B，实现 A=AUB 和 A=A⋂B.

2). 用单链表表示一元多项式，并实现输入，输出，加、减运输。

3). 将图书信息以线性表结构组织，实现一个小型的图书管理系统，（包括插入，删除，查找，输出等基本操作）

三、提示：集合 A，B 的数据分别放在线性表中，A=AUB 的思路是扫描 B 中每个元素，查看它是否在 A 中，不在则将其插入到 A 中。A=A∩B 实现方法：扫描 A 表，取出每个数据元素看该元素是否在 B 中，不在则将该元素从 A 中删除。

四、测试数据 A={2,5,18,3,45,8,9,22} B={1,4,6,9,10}

A={1,2,3,7,5}　B={4,5,7,9,1}

五、编程知识点

实验过程例子：

步骤：

1)、分析题目要求确定使用的数据结构（逻辑结构、物理结构和基本操作）

集合里面数据对象是线性关系（逻辑），考虑到求解 A=AUB 时，可以将 A 中没有的元素插入到 A 的表尾，使用顺序表比较合适，而求解 A=A⋂B 时，要将 A 中不属于 B 的元素删除，可能要删除中间的元素，使用顺序表要大量移动数据，使用单链表比较合适。两种综合，所以顺序表和链表两种存储结构各有利弊，选择那个都可以。这里使用单链表（存储）。根据题目要求可以使用的基本操作包括初始化，插入，删除，查找，遍历显示。

2)、结构定义和基本操作实现

```
//单链表结点定义
typedef struct node{
                int *data;
                struct node *next;
            } node,*link;
//单链表基本操作
status init_link(link &L); //带头结点的单链表表初始化
status insert_link(link &L,int i,ElemType e)//向表中第 i 个位置插入数据
status del_link(link &L,int i)//从表中删除第 i 个位置数据
link locate_link(link L,int e);//找和 e 相同的元素，找到返回元素所在结点地址，找不到返回空
void trave_link(link L)//遍历带头结点单链表
```

3)、其他算法设计：集合存入线性表（建立单链表），A=AUB 和 A=A⋂B 算法

```
status create_link(link &L )//建立带头结点单链表
void union(link &LA,link LB);//求集合 LA 和 LB 的并到 LA 中
void intersection(link &LA,link LB)//求两个集合的交集
```

例：A=AUB

```
void union(link &A, link B )
{
    p=B->next;
    while(p!= NULL)
    {
        x=p->data;
        if(!locate_link(A, x)) insert_link(A,1,x);
        p=p->next;
    }
}
```

A=AUB

```
void intersection(link &A, link B )
{
    p=A->next; i=1;
    while(p!= NULL)
    {
        x=p->data;
        p=p->next;
        if(!locateList(B, x))
         del_link(A,i);
        else
        i++;
    }
}
```

4)、代码实现

文件的组织：将类型的定义放入头文件中，将基本操作代码实现放入源文件中。总控信息命令放入 main 文件

5)、代码调试

编译（查找语法错误)、修改-------》no error 运行

白盒调试（查找逻辑错误，算法错误）-------》正确运行

6)、老师验收

7)、总结写报告

六、支撑课程目标 1,2

实验二 栈和队列的定义和应用

实验学时：2 学时

一、**实验目的**:

理解栈、队列的定义

掌握栈的两种存储形式下基本操作的实现

掌握队的两种存储形式下基本操作的实现

理解栈和队列的作用

二、实验内容：(选择其一完成，也可都做)

1）.（成绩最高为 B）

实现对用户输入的 10 进制数转化成其他进制并输出.

2）. (成绩最高为 A)

计算用户输入的简单表达式的值：例如（5+2）*3。即输入的表达式中数字为一位整数即可。

3）.（成绩最高为 A）

实现模拟银行排队系统

三、提示：

1）. 使用栈存储十进制数除以 m（要转成 m 进制）得到的余数。当商为 0 时结束进栈。依次出栈的数据序列就是转化成的 m 进栈数。

2）. 准备两个栈，一个存放操作数，一个存放运算符。根据严蔚民编著《数据结构》中表达式求值伪代码编写。

3）.来银行办事的人可以定义成按办事不同花费的时间不同，存钱，取钱，查询三种情况定。

四、测试数据

1）. 125 转化成二进栈 1111101　将 100 转成 8 进栈 144

2）. 3+9*1　　（3+2）*3

3）.银行原有前 1000，来客户：存钱 100，存钱 300，取钱 200，取钱 400，查询，取钱 1000，存钱 500，查看

五、编程知识点

C 程序中多个文件的组织问题。

以下内容摘自 https://www.cnblogs.com/clover-toeic/p/3728026.html，有兴趣的同学可以详细阅读。

C 语言里，每个源文件是一个模块，头文件为使用该模块的用户提供接口。接口指一个功能模块提供给其他模块用以访问具体功能的函数。使用源文件实现模块的功能，使用头文件提供单元的接口。用户只需包含相应的头文件就可使用该头文件中提供的接口。

通过头文件包含的方法将程序中的各功能模块联系起来有利于模块化程序设计:

1）通过头文件调用库功能。在很多场合，源代码不便(或不准)向用户公布，只要向用户提供头文件和二进制库即可。用户只需按照头文件中的接口声明来调用库功能，而不必关心接口如何实现。编译器会从库中提取相应的代码。

2）头文件能加强类型安全检查。若某个接口的实现或使用方式与头文件中的声明不一致，编译器就会指出错误。这一简单的规则能大大减轻程序员调试、改错的负担。

在预处理阶段，编译器将源文件包含的头文件内容复制到包含语句(#include)处。在源文件编译时，连同被包含进来的头文件内容一起编译，生成目标文件(.obj)。如果所包含的头文件非常庞大，则会严重降低编译速度(使用 GCC 的-E 选项可获得并查看最终预处理完的

文件)。因此，在源文件中应仅包含必需的头文件，且尽量不要在头文件中包含其它头文件。
头文件定义的规则：
1. 头文件的语义相关性原则：同一头文件中出现的类型定义、函数声明应该是语义相关的、有内部逻辑关系的，避免将无关的定义和声明放在一个头文件中。
2. 头文件名应尽量与实现功能的源文件相同，即 module.c 和 module.h。但源文件不一定要包含其同名的头文件。
3. 头文件内不允许定义变量和函数，只能有宏、类型(typedef/struct/union/enum 等)及变量和函数的声明。特殊情况下可 extern 基本类型的全局变量，源文件通过包含该头文件访问全局变量。
4.头文件定义样式：

```
#ifndef   _PRJ_DIR_FILE_H   //必须确保 header guard 宏名永不重名
#define   _PRJ_DIR_FILE_H

//<头文件内容>

#endif
```

5. 减少头文件的嵌套和交叉引用，头文件仅包含其真正需要显式包含的头文件。

例如，头文件 A 中出现的类型定义在头文件 B 中，则头文件 A 应包含头文件 B，除此以外的其他头文件不允许包含。

头文件的嵌套和交叉引用会使程序组织结构和文件组织变得混乱，同时造成潜在的错误。大型工程中，原有头文件可能会被多个其他(源或头)文件包含，在原有头文件中添加新的头文件往往牵一发而动全身。若头文件中类型定义需要其他头文件时，可将其提出来单独形成一个全局头文件。

例：

将复杂的程序根据它们的功能设计成由几个源文件和头文件构成。以本实验的 b 为例子。实现计算用户输入的简单表达式的值。

stack.h 内容：

```
#ifndef MY_STACK1
#define MY_STACK1
#include “mymain.h”
#define SIZE 100
typedef int stackelemtype;
typedef struct {
    stackelemtype *top, *base ;
    int size;
}Stack; //整数栈类型的定义
//下面是栈的基本操作函数声明
status initstack(Stack &s); //整数栈的初始化
status push(Stack &s, stackelemtype e); //整数栈的进栈
status pop (Stack &s, stackelemtype &e); //整数栈的出栈
status empty (Stack s); //整数栈的判空
status full(Stack s); //整数栈的判满
stackelemtype gettop(Stack s); //整数栈的取栈顶元素
```

```
#endif
```

stack.cpp 内容：

```
#include " stack.h"
status initstack(Stack &s) //整数栈的初始化
{//函数体
}
status push(Stack &s, stackelemtype e)//整数栈的进栈
{//函数体
}

status pop (Stack &s, stackelemtype &e) //整数栈的出栈
{//函数体
}

status empty (Stack s) //整数栈的判空
{//函数体
}

status full(Stack s) //整数栈的判满
{//函数体
}

stackelemtype gettop(Stack s) //整数栈的取栈顶元素
{//函数体
}
```

expression.h 内容：

```
#ifndef MY_EXPRESSION
#define MY_ EXPRESSION

char priority(char ch1, char ch2);//判断栈顶运算符 ch1 与栈外运算符 ch2 的优先级
int calculate(int num1, char ch, int num2);//num1  与 num2 进程 ch 运算，结果返回
void cal_express();//进行表达式求值的主体函数

#endif
```

expression.cpp 内容：

```
#include "stack.h"
#include "expression.h"
char priority(char ch1, char ch2) //判断栈顶运算符 ch1 与栈外运算符 ch2 的优先级
{
//函数体
}
```

```
int calculate(int num1, char ch, int num2)//num1 与 num2 进程 ch 运算，结果返回
{
//函数体
}

void cal_express()//进行表达式求值的主体函数
{
//函数体
}
```

mymain.h 内容:

```
#ifndef MY_MAINEXP2
#define MY_ MAINEXP2
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define OK 1
# define ERROR 0
typedef int status;

#endif
```

mymain.cpp 内容：

```
#include “mymain.h”
#include “expression.h”
int main ()
{
//函数体
cal_express(); //调用计算表达式求值的函数
return 0;
}
```

六、支撑课程目标 1, 2

实验三 二叉树及其基本操作

实验学时：4 学时

一、实验目的：

掌握二叉树的逻辑结构和常用存储结构

掌握二叉树的三种遍历及其应用

二、实验内容：

1). 定义二叉链表的存储结构

2). 编写下面的函数

a.建立一棵二叉树

b.实现二叉树的三种递归的遍历

c.实现二叉树的先根，中根非递归遍历

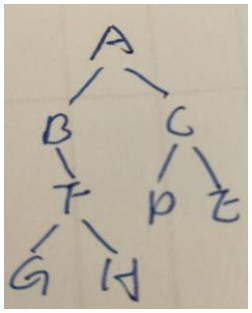
d.求二叉树叶子结点的个数

e.求二叉树的深度(选做)

f.按层遍历二叉树(选做)

三、提示：使用栈进行先根和中根非递归时，栈内数据类型为指向二叉树的节点的指针。进行按层遍历时使用队列，队列内的元素也是指向二叉树节点的指针类型。

四、测试数据：建立如下二叉树进行测试。



AB#FG##H##CD##E##

五、知识点：程序调试

1.在没有语法错误，运行确不正确时，先查看每个文件或函数，看有没有逻辑或书写错。如果实在找不出问题所在，可以使用加断点调试的方法。在你认为可能有问题的函数中加入断点，然后单步调试，查看变量的变化，寻找问题原因。

六、支撑课程目标 1, 2

实验四 图及其基本操作

实验学时：4 学时

一、实验目的：

掌握图的定义

掌握图的邻接矩阵和邻接表存储形式

掌握图的深度优先，广度优先遍历，求最小生成树和最短路径，关键路径，拓扑排序等算法

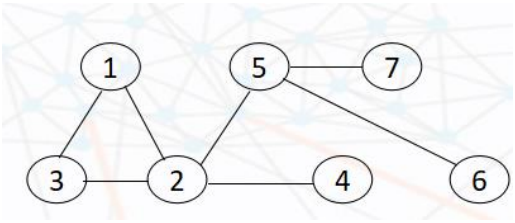
二、实验内容：

1) 将图以邻接表或邻接矩阵的形式存储
2) 图的深度优先，广度优先遍历
3) 求一个连通图的最小生成树(选做)
4) 求一个图中给定点到其他各个顶点的最短路径(选做)
5) 求一个 AOE 网的关键路径(选做)
6) 求一个 AOV 网的拓扑排序(选做)

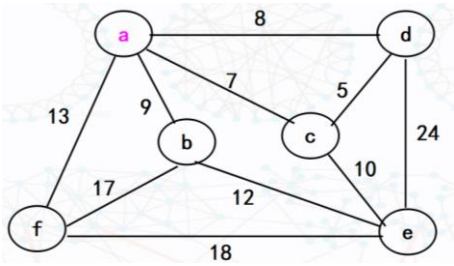
三、提示：图的深度优先遍历和广度优先遍历、拓扑排序和关键路径可以在邻接链表的存储形式下实现，最小生成树和最短路径可以在邻接矩阵的存储形式下进行。

四、测试数据：

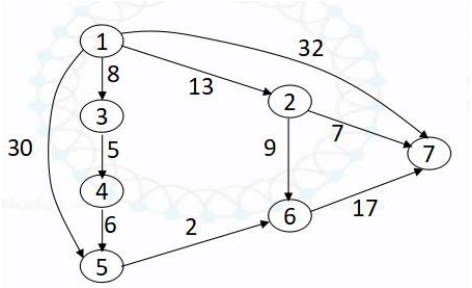
2)测试图



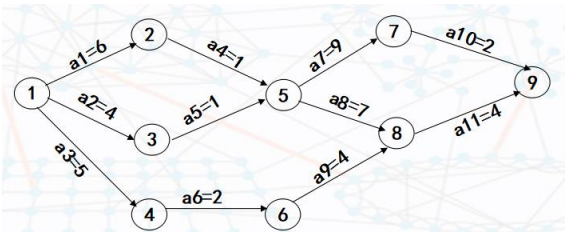
3）测试图



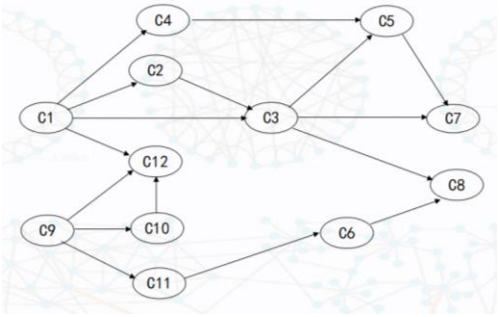
4）测试图



5）测试图



6）测试图



五、支撑课程目标 1, 2

实验五 查找和排序

实验学时：4 学时

一、实验目的:

掌握顺序查找和折半查找；掌握索引查找

掌握几种常用排序算法并掌握他们的特点；掌握各排序方法的时间复杂度和空间复杂度;

二、实验内容：

1）.

a).建立如课本所示索引表（数据量再大一些）

b).对用户输入的数据进行查找。在索引表中查找用折半查找，在子表中查找用顺序查找

2）. 实现直接插入排序；冒泡排序；简单选择排序；快速排序；希尔排序(选做)；堆排序(选做)

以上排序的数据可以是一批整数，整数最好从文件导入。

三、支撑课程目标 3

附录 A

河北地质大学

课程实验报告

课程名称：数据结构实验

专业和班级：20 级软件工程 1 班

姓名：

学号：

报告日期：

信息工程学院

实验二 栈和队列的定义和应用

1. 实验目的:

理解栈、队列的定义
掌握栈的两种存储形式下基本操作的实现
掌握队的两种存储形式下基本操作的实现
理解栈和队列的作用

2. 实验内容:

实现利用队列来模拟银行办理业务的过程。客户业务分为两种。第一种是申请从银行得到一笔资金，即取款或借款。第二种是向银行投入一笔资金，即存款或还款。银行有两个服务窗口，相应地有两个队列。客户到达银行后先排第一个队。处理每个客户业务时，如果属于第一种，且申请额超出银行现存资金总额而得不到满足，则立刻排入第二个队等候，直至满足时才离开银行；否则业务处理完后立刻离开银行。每接待完一个第二种业务的客户，则顺序检查和处理(如果可能)第二个队列中的客户，对能满足的申请者予以满足，不能满足者重新排到第二个队列的队尾。注意，在此检查过程中，一旦银行资金总额少于或等于刚才第一个队列中最后一个客户(第二种业务)被接待之前的数额，或者本次已将第二个队列检查或处理了一遍，就停止检查(因为此时已不可能还有能满足者)转而继续接待第一个队列的客户。任何时刻都只开一个窗口。假设检查不需要时间。营业时间结束时所有客户立即离开银行。

写一个上述银行业务的事件驱动模拟系统，通过模拟方法求出客户在银行内逗留的平均时间。

3. 对应课程目标：课程目标 1，课程目标 2

4. 分析问题，设计所用数据结构并用 ADT 表示

实验要求可知，本实验主要使用的数据结构是队列。队列的抽象数据类型如下:

ADT Queue{
数据对象：D={ai|ai∈ElemSet,i=1,2, ……,n,n>=0}
数据关系：R={<ai-1,ai>| ai-1,ai∈D,i=2,3, ……,n}
基本操作:
CreateList（&L,close_time）
初始条件：队列的链式表和 close_time 已存在。
操作结果：构造一个队列客户信息事件表 L。
init_Queue(&Q)
操作结果：构造一个空队列 Q。
EmptyQueue (Q)
初始条件：队列 Q 已存在。
操作结果：若 Q 为空队列，则返回 TRUE，否则返回 FALSE。
EnterQueue(&Q, e)
初始条件：队列 Q 已存在。

操作结果：插入元素 e 为 Q 的新的队尾元素。

OutQueue(&Q, &e)

初始条件：队列 Q 已存在且非空。

操作结果：删除 Q 的队头元素，并用 e 返回其值。

Wait_Queue(L, &Q, &S, current_amount, min_amount)

初始条件：队列的链式表 L、队列 Q，S、银行当前资金总额 current_amount、客户取款的最小金额 min_amount 已存在。

操作结果：处理能办理业务的客户，办理完后客户离开银行，否则客户排到队尾等待

Deal_Queue(L,&P,&Q,&S,total)

初始条件：队列的链式表 L、队列 P、Q、S、银行当前资金总额 total 已存在。

操作结果：将数据按照是否能够满足需求进行分类。

OutPut(L, S, total)

初始条件：队列的链式表 L、队列 S、银行当前资金总额 total 已存在。

操作结果：输出所有信息

}ADT Queue;

5. C 语言描述所用数据结构，基本操作的函数声明

我使用链式存储来实现队列的基本操作，队列结点定义：

```
typedef struct event
{
        char name[20];          //客户名
        int amount ;          //存取款金额
        int slove_time;         //处理需要的时间
        int arrive_time;    //到达时间(距开业的分钟数)
        int interval_time;  //与前一个客户的间隔时间
        int wait_time;      //总的等待时间
}Elemtype;
typedef struct Node
{
        Elemtype data;
        struct Node *next;
}Node,*LinkList;
typedef struct
{
    LinkList front,rear ;
}QUEUE;
```

基本操作的函数声明:

```
int CreateList(LinkList &L,int close_time)；//建立事件表
int init_Queue(QUEUE &Q)；//队列初始化
int EmptyQueue(QUEUE Q)；//判断是否为空队列
int EnterQueue(QUEUE &Q,Elemtype e)；//入队
Elemtype OutQueue(QUEUE &Q,Elemtype &e)；//出队
void Wait_Queue(LinkList L,QUEUE &Q,QUEUE &S,int current_amount,int
```

min_amount)；//等待处理函数
void Deal_Queue(LinkList L,QUEUE &P,QUEUE &Q,QUEUE &S,int total);// 处理客户需求函数
void OutPut(LinkList L,QUEUE S,int total)；//输出函数

6. 源代码链接 stack

7. 心得体会

我的收获是。。。。。。。。

8. 评语和成绩

附录 B

	评阅标准	得分
教师评阅	独立完成实验要求中 A 等题目，正确回答老师针对实验内容提出的关于数据结构，算法和程序的问题，通过教师验收，数据结构和算法掌握良好。实验报告书写逻辑清楚，数据结构设计合理，报告规范。	A
	独立完成实验要求中 B 等题目，通过教师验收，教师验收时并回答问题准确，算法掌握良好。实验报告规范。或者，他人协助下完成实验要求中 A 等题目，通过教师验收，教师验收时并回答问题不准确，数据结构或算法有模糊的地方。实验报告规范。	B
	在他人协助下完成实验，通过教师验收，教师验收时能正确回答 60%以上的问题，数据结构和算法基本掌握。实验报告规范。	C
	在他人协助下完成实验，教师验收时不能正确回答问题，算法不清楚，未通过教师验收，或没有完成实验或未提交实验报告。	D