《算法与数据结构》课程设计任务书

一、设计任务

设计一个应用程序，利用多级菜单实现线性表、栈、队列、二叉树及图五种结构的基本操作及应用。具体内容包括：

1．线性表的基本操作及应用
   ① 创建
   ② 插入
   ③ 删除
   ④ 查找
   ⑤ 遍历
   ⑥ 应用
2．栈的基本操作及应用
   ① 进栈
   ② 出栈
   ③ 取栈顶元素
   ④ 应用
3．队列的基本操作及应用
   ① 入列
   ② 出列
   ③ 取队头元素
   ④ 取队尾元素
   ⑤ 应用
4．二叉树的基本操作及应用
   ① 创建
   ② 遍历
   ③ 找双亲
   ④ 找兄弟
   ⑤ 找孩子
   ⑥ 应用
5．图的基本操作及应用
   ① 创建
   ② 深度优先遍历
   ③ 广度优先遍历
   ④ 找第一个邻接点
   ⑤ 找下一个邻接点
   ⑥ 求顶点的度
   ⑦ 应用

二、设计要求

1）存储结构

①线性表使用链式存储结构，如单链表。

②栈使用顺序存储结构，如顺序栈。

③队列的存储结构不限，链队列或循环队列均可。

④二叉树使用链式存储结构，如二叉链表。

⑤图的存储结构不限，邻接矩阵或邻接表均可。

2）应用选题

①**历史记录**。浏览器存储用户访问的网页历史，包括保存网页信息，同时形成访问链，支持前进/后退功能。设计一个应用程序，选择合适的数据结构实现以上功能。

②**播放列表**。在音乐播放器的播放列表中，用户可以顺序播放列表里的歌曲，也可以向列表中添加歌曲、删除歌曲等。设计一个应用程序，选择合适的数据结构实现以上功能。

③**服务调度**。平台用户完成订单支付后，订单服务发送“支付成功通知”，为防止宕机丢失，需持久化存储消息，同时推送消息给库存服务通知扣减库存，推送消息给通知服务发送短信给用户等。各服务接收消息并成功完成后，回送确认消息给发送服务。为提高效率，各服务异步处理、解耦，即相互之间不直接调用，避免一方修改影响另一方；同时各自异步处理，提高响应速度。设计一个应用程序，选择合适的数据结构完成以上任务。

④**类型检查**。在表达式解析中，可构建抽象语法树来检查表达式类型。例如，对于表达式 x*(y+z)，构建一棵运算符为内部结点，操作数为叶子结点的语法树，再通过自底向上的方式推导子树类型，如 int 类型的 y 和 float 类型的 z 相加，得到值的类型为 float 类型。

⑤**社交推送**。在社交平台中，用户之间可以相互关注，但用户 A 关注用户 B，用户 B 不一定关注用户 A。其中，用户被关注的数量为用户的传播力；用户关注他人的数量为用户的关注数。显然，被高传播力关注的用户将有助于自身传播力的提升。设计一个应用程序，选择合适的数据结构实现向用户推送高传播力用户列表。

注：利用所学数据结构完成以上若干应用选题（模拟），并添加到对应结构的应用模块中。

3）程序要求

①整合课内上机所完成的各类结构的基础操作，完成若干新添加的结构应用。

②所有二级菜单中的基础操作可根据应用需求扩展，原则上不少于所列出的操作。

③程序独立完成，运行正确，无编译错误，无逻辑错误。

④应用选题为可选任务，原则上任务完成越多，得分越高。

4）**课设报告要求** 报告格式规范，语言流畅，功能实现描述清楚，测试设计合理，

结论准确。具体内容包括：
①设计方案；
②实现过程；
③实现代码；
④测试与结论；
⑤难点与收获。

三、设计指导（参考）

```
#include<stdio.h>
void ShowMainMenu(){
printf("\n");
printf("**********数据结构**********\n");
printf("* 1   线性表的基本操作及应用                    *\n");
printf("* 2   栈的基本操作及应用                        *\n");
printf("* 3   队列的基本操作及应用                      *\n");
printf("* 4   二叉树的基本操作及应用                    *\n");
printf("* 5   图的基本操作及应用                        *\n");
printf("* 6   退出                                      *\n");
printf("************************************************\n");
}
void LinkList(){
int n;
do{
printf("\n");
printf("**********线性表的基本操作及应用**********\n");
printf("* 1   创建                                      *\n");
printf("* 2   插入                                      *\n");
printf("* 3   删除                                      *\n");
printf("* 4   查找                                      *\n");
printf("* 5   遍历                                      *\n");
printf("* 6   应用                                      *\n");
printf("* 7   退出                                      *\n");
printf("************************************************\n");
printf("请选择：");
scanf("%d",&n);
switch(n){
case 1:
    printf("--------创建表---------");break;
case 2:
    printf("--------插入一个元素-------");break;
case 3:
    printf("--------删除一个元素-------");break;
case 4:
```

```
        printf("--------查找一个元素-------");break;
case 5:
        printf("--------输出所有元素---------");break;
case 6:
        printf("--------应用---------");break;
case 7: break;
default:
        printf("ERROR!");break;
}
}while(n!=7);
}
void Stack(){
int n;
do{
printf("\n");
printf("*****************栈的基本操作及应用*****************\n");
printf("* 1    进栈                                          *\n");
printf("* 2    出栈                                          *\n");
printf("* 3    取栈顶元素                                    *\n");
printf("* 4    应用                                          *\n");
printf("* 5    退出                                          *\n");
printf("*****************************************************\n");
printf("请选择：");
scanf("%d",&n);
switch(n){
case 1:
        printf("--------进栈-------");break;
case 2:
        printf("--------出栈-------");break;
case 3:
        printf("--------取栈顶元素-------");break;
case 4:
        printf("--------应用-------");break;
case 5:break;
default:
        printf("ERROR!");break;
}
}while(n!=5);
}
void Queue(){
int n;
do{
printf("\n");
printf("*************队列的基本操作及应用*************\n");
```

```
printf("* 1    入列                                                *\n");
printf("* 2    出列                                                *\n");
printf("* 3    取队头元素                                          *\n");
printf("* 4    取队尾元素                                          *\n");
printf("* 5    应用                                                *\n");
printf("* 6    退出                                                *\n");
printf("**************************************************\n");
printf("请选择：");
scanf("%d",&n);
switch(n){
case 1:
    printf("---------入列-------");break;
case 2:
    printf("---------出列-------");break;
case 3:
    printf("---------取队头元素-------");break;
case 4:
    printf("---------取队尾元素-------");break;
case 5:
    printf("---------应用-------");break;
case 6:break;
default:
    printf("ERROR!");break;
}
}while(n!=6);
}
void BiTree(){
int n;
do{
printf("\n");
printf("**************二叉树的基本操作及应用**************\n");
printf("* 1    创建                                                *\n");
printf("* 2    遍历                                                *\n");
printf("* 3    查找双亲                                            *\n");
printf("* 4    查找兄弟                                            *\n");
printf("* 5    查找孩子                                            *\n");
printf("* 6    应用                                                *\n");
printf("* 7    退出                                                *\n");
printf("******************************************************\n");
printf("请选择：");
scanf("%d",&n);
switch(n){
case 1:
    printf("---------创建二叉树--------");break;
```

```
case 2:
    printf("---------遍历（先/中/后）-------");break;
case 3:
    printf("---------查找双亲-------");break;
case 4:
    printf("---------查找兄弟（左/右）-------");break;
case 5:
    printf("---------查找孩子（左/右）-------");break;
case 6:
    printf("---------应用-------");break;
case 7:break;
default:
    printf("ERROR!");break;
}
}while(n!=7);
}
void Graph(){
int n;
do{
printf("\n");
printf("****************图的基本操作及应用****************\n");
printf("* 1   创建                                          *\n");
printf("* 2   深度优先遍历                                  *\n");
printf("* 3   广度优先遍历                                  *\n");
printf("* 4   找第一个邻接点                                *\n");
printf("* 5   找下一个邻接点                                *\n");
printf("* 6   求顶点的度                                    *\n");
printf("* 7   应用                                          *\n");
printf("* 8   退出                                          *\n");
printf("****************************************************\n");
printf("请选择：");
scanf("%d",&n);
switch(n){
case 1:
    printf("---------创建图（有向/无向）-------");break;
case 2:
    printf("---------深度优先遍历-------");break;
case 3:
    printf("---------广度优先遍历-------");break;
case 4:
    printf("---------找第一个邻接点-------");break;
case 5:
    printf("---------找下一个邻接点-------");break;
case 6:
```

```
        printf("---------求顶点的度-------");break;
case 7:
        printf("---------应用-------");break;
case 8:break;
default:
        printf("ERROR!");break;
}
}while(n!=8);
}

int main(){
int n;
do{
ShowMainMenu();
printf("请选择：");
scanf("%d",&n);
switch(n){
case 1:LinkList();break;
case 2:Stack();break;
case 3:Queue();break;
case 4:BiTree();break;
case 5:Graph();break;
case 6:break;
default:printf("ERROR!");break;
}
}while(n!=6);
return 1;
}
```