数据结构试验报告

专业：计算机科学与技术

姓名：

学号：

实验一　线性表的应用

【实验目的】

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. 以线性表的各种操作（建立、插入、删除、遍历等）的实现为重点；
3. 掌握线性表的动态分配顺序存储结构的定义和基本操作的实现；
4. 通过本章实验帮助学生加深对C 语言的使用（特别是函数的参数调用、指针类型的应用和链表的建立等各种基本操作）。

【实验内容】

约瑟夫问题的实现：n 只猴子要选猴王，所有猴子按 1,2，„，n 编号围坐一圈，从第 1 只开始按 1,2，„，m 报数，凡报到m 号的猴子退出圈外，如此循环报数，直到圈内剩下一只猴子时，这个猴子就是猴王。编写一个程序实现上述过程，n 和m 由键盘输入。

【实验要求】

1. 要求用顺序表和链表分别实现约瑟夫问题；

2. 独立完成，严禁抄袭；

3. 上交的实验报告由如下部分组成：①实验名称②实验目的③实验内容（问题描述，算法描述，程序清单，测试结果，算法分析）

【程序实现】

顺 序 表 ：

```
#include<stdio.h>
#define MaxSize 100
void jose(int n,int m) {
	int mon[MaxSize];
	int i,d,count;
	for(i=0;i<n;i++)
		mon[i]=i+1;
	printf("出队前： ");
	for(i=0;i<n;i++)
		printf("%d\t",mon[i]);
	printf("\n");
	printf("出队序列： ");
	count=0;
	i=-1;
	while (count<n)  {
		d=0;
		while (d<m)    {
			i=(i+1)%n;
			if(mon[i]!=0)
				d++;
		}
```

```
            printf("%d\t",mon[i]);
            mon[i]=0;
            count++;
        }
        printf("\n");
}
void main() {
        int n,m;
        printf("输入猴子数！\n");
        scanf("%d",&n);
        printf("输入报号退出数！\n");
        scanf("%d",&m);
        jose(n,m);
}
```

链表：

```
#include    <stdio.h>
#include   <stdlib.h>
typedef struct node
{
        int data;
        struct node *next;
}linklist;
int main()
{
        int i, n, k, m, total;
        linklist *head, *p, *s,  *q;
        printf("请输入猴子的个数:");
        scanf("%d", &n);
        printf("请输入要从第几个猴子开始报数:");
        scanf("%d", &k);
        printf("请输入出局数字:");
        scanf("%d", &m);
  head = (linklist*) malloc(sizeof(linklist));
  p = head;
  p->data = 1;
  p->next = p;
  for (i = 2; i <= n; i++){
          s = (linklist*) malloc(sizeof(linklist));
          s->data = i;
          s->next = p->next;
          p->next = s;
          p = p->next;
  }
p = head;
```

```
for (i = 1; i < k; i++){
        p = p->next;
}
total = n;
printf("\n 出局序列为:");
q = head;
while (total != 1){
        for (i = 1; i < m; i++){
                p = p->next;
        }
        printf("[%d] ", p->data);
        while (q->next != p){
                q = q->next;
        }
                q->next = p->next;
                s = p;
                p = p->next;
                free(s);
                total--;
        }
        printf("\n\n 猴子大王为第  [%d] 号\n\n", p->data);
        free(p);
}
```

实验二　栈和队列的应用

【实验目的】

1. 熟练掌握栈和队列的结构，以及这两种数据结构的特点；
2. 能够在两种存储结构上实现栈的基本运算，特别注意栈满栈空的判断条件和描述方法；
3. 熟练掌握链队列和循环队列的基本运算，特别注意队列满和队列空的判断条件和描述方法。

【实验内容】

表达式求值的实现：输入一个包含“+”、“-”、“*”、“/”、正整数和圆括号的合法表达式，用算符优先法计算该表达式的结果。

【实验要求】

1. 要求用栈实现表达式求值问题；
2. 独立完成，严禁抄袭；
3. 上交的实验报告由如下部分组成：①实验名称②实验目的③实验内容（问题描述，算法描述，程序清单，测试结果，算法分析）

【程序实现】

```
#include<stdio.h>
```

```
#include<stdlib.h>
#define MaxOp 128
#define MaxSize 128
#define Size 128

struct {
	char ch; //运算符
	int pri; //优先级
}
lpri[]={{'=',0},{'(',1},{'*',5},{'/',5},{'+',3},{'-',3},{')',6}},
rpri[]={{'=',0},{'(',6},{'*',4},{'/',4},{'+',2},{'-',2},{')',1}};
int leftpri(char op) {
	int i;
	for(i=0;i<MaxOp;i++)
		if(lpri[i].ch==op)
			return lpri[i].pri;
}
int rightpri(char op) {
	int i;
	for(i=0;i<MaxOp;i++)
		if(rpri[i].ch==op)
			return rpri[i].pri;
}
int InOp(char ch) {
	if(ch=='('||ch==')'||ch=='+'||ch=='-'||ch=='*'||ch=='/')
		return 1;
	else
		return 0;
}
int Precede(char op1,char op2) {
	if(leftpri(op1)==rightpri(op2))
		return 0;
	else if(leftpri(op1)<rightpri(op2))
		return -1;
	else return 1;
}
void trans(char *exp,char postexp[]){
	struct{
		char data[MaxSize]; //存放运算符
		int top; //栈指针
	}op; //定义运算符栈
	int i=0; //i 作为  postexp 的下标
	op.top=-1;
	op.top++; //将'='进栈
```

```
op.data[op.top]='=';
while(*exp!='\0') {
	if(!InOp(*exp)) {
		while(*exp>='0'&&*exp<='9') {
			postexp[i++]=*exp;
			exp++;
		}
		postexp[i++]='#'; //用#标志一个数值串的结束
	}
	else
		switch(Precede(op.data[op.top],*exp)){
case -1: //栈顶运算符的优先级低
	op.top++;op.data[op.top]=*exp;
	exp++; //继续扫描其他字符
		break;
	case 0: // 只有括号满足这种情况
		op.top--; //将退栈
		exp++;
		break;
	case 1:
		postexp[i++]=op.data[op.top];
		op.top--;
		break;
		}
	}
	while(op.data[op.top]!='=') {
		postexp[i++]=op.data[op.top];
		op.top--;
	}
	postexp[i]='\0';
}

float compvalue(char *postexp){
	struct	{
		float data[MaxSize]; //存放数值
		int top; //栈指针
	}st;
	float d,a,b,c;
	st.top=-1;
	while(*postexp!='\0')  {
		switch(*postexp){
		case '+':
			a=st.data[st.top];
			st.top--;
```

```
            b=st.data[st.top];
            st.top--;
            c=a+b;
            st.top++;
            st.data[st.top]=c;
            break;
        case '-':
            a=st.data[st.top];
            st.top--;
            b=st.data[st.top];
            st.top--;
            c=b-a;
            st.top++;
            st.data[st.top]=c;
            break;
        case '*':
            a=st.data[st.top];
            st.top--;
            b=st.data[st.top];
            st.top--;
            c=a*b;
            st.top++;
            st.data[st.top]=c;
            break;
        case '/':
            a=st.data[st.top];
            st.top--;
            b=st.data[st.top];
            st.top--;
            if(a!=0){
                c=b/a;
                st.top++;
                st.data[st.top]=c;
            }
            else{
                printf("\n\t 除零错误！  \n");
                exit(0);
            }
            break;
        default:
            d=0;
            while(*postexp>='0'&&*postexp<='9'){
                d=10*d+*postexp-'0';
                postexp++;
```

```
                }
                st.top++;
                st.data[st.top]=d;
                break;
            }
            postexp++;
        }
        return(st.data[st.top]);
}
 void main(){
        for(;;){
            char exp[Size];
            char postexp[MaxSize];
            printf("请输入表达式:");
            gets(exp);
            trans(exp,postexp);
            printf("中缀表达式: %s\n",exp);
            printf("后缀表达式: %s\n",postexp);
            printf("表达式的值: %g\n",compvalue(postexp));
      }
}
```

实验三　数组的应用

【实验目的】

1. 掌握数组的两种存储表示方法；
2. 掌握对特殊矩阵进行压缩存储时的下标变换公式；
3. 掌握稀疏矩阵的两种压缩存储方法的特点和适用范围。

【实验内容】

稀疏矩阵转置的实现：用三元组顺序表做存储结构，实现稀疏矩阵的转置。

【实验要求】

1. 已知某一稀疏矩阵的三元组顺序表，由其直接得到其转置矩阵的三元组顺序表；

2. 独立完成，严禁抄袭；

3. 上交的实验报告由如下部分组成：①实验名称②实验目的③实验内容（问题描述，算法描述，程序清单，测试结果，算法分析）

【程序实现】

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
```

```
#include<ctype.h>
typedef int data;

struct tuple3tp    {
        int i,j;
        int v;
};

struct sparmattp{
        int mu,nu,tu;
        struct tuple3tp data[31];
};

struct sparmattp a,b;
void crt_sparmat() {
        int i;
        printf("输入稀疏矩阵行值，列值，最大非零元个数：");
        scanf("%d%d%d",&a.mu,&a.nu,&a.tu);

        for(i=1;i<=a.tu;i++){
                printf("输入行坐标，列坐标，非零元");
                scanf("%d%d%d",&a.data[i].i,&a.data[i].j,&a.data[i].v);
        }
}

void trans_sparmat() {
        int col,p,q;
        b.mu=a.mu;
        b.nu=a.nu;
        b.tu=a.tu;
        if(b.tu!=0)    {
                q=1;
                for(col=1;col<=a.nu;col++)
                        for(p=1;p<=a.tu;p++)
                                if(a.data[p].j==col){
                                        b.data[q].i=a.data[p].j;
                                        b.data[q].j=a.data[p].i;
                                        b.data[q].v=a.data[p].v;
                                        q++;
                                }
        }
}

void out(struct sparmattp x)  {
```

```
int i,j,k,flag;
for(i=1;i<=x.mu;i++)    {
        for(j=1;j<=x.nu;j++)      {
                flag=0;
                for(k=1;k<=x.tu;k++)        {
                        if(((x.data[k].i)==i)&&((x.data[k].j)==j))          {
                                flag=1;
                                printf("%5d",x.data[k].v);
                        }
                }
                if(flag==0)
                        printf(" 0");
        }           printf("\n");
}
}

void main() {
        printf("稀疏矩阵的建立与转置\n");
        crt_sparmat();
        trans_sparmat();
        printf("原矩阵为：\n");
        out(a);
        printf("转置矩阵为：\n");
        out(b);
}
```

实验四　树和二叉树的应用

【实验目的】

1. 熟练掌握树的基本概念、二叉树的基本操作及在链式存储结构上的实现；
2. 重点掌握二叉树的生成、遍历及求深度等算法；
3. 掌握哈夫曼树的含义及其应用。
4. 掌握运用递归方式描述算法及编写递归C 程序的方法，提高算法分析和程序设计能力。

【实验内容】

二叉树采用二叉链表作存储结构，试编程实现二叉树的如下基本操作：

1. 按先序序列构造一棵二叉链表表示的二叉树T；
2. 对这棵二叉树进行遍历：先序、中序、后序以及层次遍历序列，分别输出结点的遍历序列；
3. 求二叉树的深度/叶结点数目。

【实验要求】

上交实验报告，要求同上。

【程序实现】

```
#include "stdio.h"
#include "string.h"
#define NULL 0
  typedef struct BiTNode{
        char data;
        struct BiTNode *lchild,*rchild;
  }BiTNode,*BiTree;
  BiTree Create(BiTree T){
        char ch;
        ch=getchar();
        if(ch=='&')
              T=NULL;
        else{
              if(!(T=(BiTNode *)malloc(sizeof(BiTNode))))
                    printf("Error!");
              T->data=ch;
              T->lchild=Create(T->lchild);
              T->rchild=Create(T->rchild);
        }
        return T;
}
void Preorder(BiTree T){
        if(T){
              printf("%c",T->data);
              Preorder(T->lchild);
              Preorder(T->rchild);
        }
}
int Sumleaf(BiTree T){
        int sum=0,m,n;
        if(T){
              if((!T->lchild)&&(!T->rchild))
                    sum++;
              m=Sumleaf(T->lchild);
              sum+=m;
              n=Sumleaf(T->rchild);
              sum+=n;
        }
        return sum;
}
void zhongxu(BiTree T){
        if(T){
              zhongxu(T->lchild);
```

```
            printf("%c",T->data);
            zhongxu(T->rchild);
        }
}
void houxu(BiTree T){
        if(T){
            houxu(T->lchild);
            houxu(T->rchild);
            printf("%c",T->data);
        }
}
int Depth(BiTree T){
        int dep=0,depl,depr;
        if(!T) dep=0;
            else{
                depl=Depth(T->lchild);
                depr=Depth(T->rchild);
                dep=1+(depl>depr?depl:depr);
                }
        return dep;
}
  int main(){
        BiTree T;
        int  sum,dep;
        printf("键盘输入:");
        printf("\n");
        T=Create(T);
        printf("先序:");
        Preorder(T);
        printf("\n");
        printf("中序:");
        zhongxu(T);
        printf("\n");
        printf("后序:");
        houxu(T);
        printf("\n");
        sum=Sumleaf(T);
        printf("和:");
        printf("%d\n",sum);
        dep=Depth(T);
        printf("深度:");
        printf("%d\n",dep);
}
```