# 数据结构

## 实验报告

| | |
|---|---|
| 学　　号： | 20189169 |
| 姓　　名： | 唐义鸿 |
| 提交日期： | 2019-12-03 |
| 成　　绩： | |

东北大学秦皇岛分校

计算机与通信工程学院

# 目录

实验一：线性表-约瑟夫环..... 2

实验二：栈-表达式求值..... 3

实验三：队列-猴子分桃..... 4

实验四：字符串与数组-三元组表稀疏矩阵.....5

实验五：树-层序遍历.....8

实验六：图-广度优先搜索..... 9

实验七：查找-二叉排序树..... 10

实验八：排序-快速排序..... 12

# 实验一：线性表-约瑟夫环

**代码：**

```
void JosephRing(){
    LinkList L,p;
    int m = 20;
    int n;
    int a[40];
    int xu;
    cout<<"请输入人数(7):" <<endl;
    cin >> n;
    cout<<"请输入这些人的密码(3,1,7,2,4,8,4):" <<endl;
    for (int i = 1; i <= n; i++ ){
        cin >> a[i];
    }
    InitList(L,n,a);
    cout<<"出列顺序为:"<<endl;
    p = L;
    int i = 1;
    while(n){
        if(i == m-1){
            Delete(p, xu, m); //删除的是p->next
            cout<<xu<<"    ";
            n--;
            i = 0;
        }
        else{
            p = p->next;
            i++;
        }
    }
}
```

**结果：**

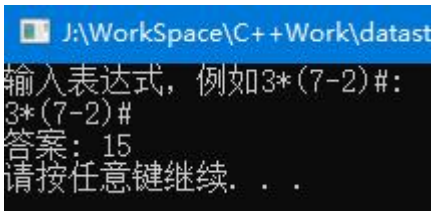
# 实验二：栈-表达式求值

**代码：**

```
void EvaluateExpression(){
    SqStack OPTR,OPND;
    InitStack(OPTR);    //OPTR是运算符栈
    InitStack(OPND);    //OPND是运算数栈
    cout<<"输入表达式，例如3*(7-2)#:"<<endl;
    Push(OPTR,'#');
    char c = getchar();

    while(c!='#' || GetTop(OPTR) != '#'){
        if (!In(c))     //判断c是否是运算符，不是则进OPND栈
        {
            Push(OPND,c);
            c = getchar();
        }
        else {
            switch(Precede(GetTop(OPTR),c)){
                case '<':  //栈顶元素优先权较低
                    Push(OPTR,c);c = getchar();
                    break;
                case '=':   //脱括号并接受下一字符
                    char x;
                    Pop(OPTR,x);
                    c = getchar();
                    break;
                case '>':  //退栈并将运算结果入栈
                    char theta,a,b;
                    Pop(OPTR,theta);
                    Pop(OPND,b);
                    Pop(OPND,a);
                    Push(OPND, Operate(a, theta, b) + 48);
                    break;
            }
        }
    }
    printf("答案: %d\n", GetTop(OPND)-0-48);//ascii码中数字与字符的转换
}
```

**结果：**



# 实验三：队列-猴子分桃

**代码：**

```
void peach(int n,int k,int m){
    //主函数 中设置 n = 5, k = 3, m = 40;
    //分别为猴的数量，筐的容量，桃的总数
    SqQueue X, P;
    InitQueue(X);
    InitQueue(P);
    int px = 0,pc = 0; //筐里的桃子,要放的桃子
    for (int i = 1; i <= n;i++){
        EnQueue(X, i);
        EnQueue(P, 0);
    }
    cout<<"答案:"<<endl;
    while(!QueueEmpty(X)){
        int x, p;
        DeQueue(X, x);
        DeQueue(P, p);

        if(pc == k)
            pc = 0;
        pc++;
        px = px + pc;

        if(px + p >= m){
            px = px - (m - p);
            p = m;

        }
        else
        {
            p = p + px;
            px = 0;
        }

        if(p == m){
            cout << x << " ";
        }
        else
        {
            EnQueue(X, x);
            EnQueue(P, p);
        }
    }
}
```

**结果：**

```
J:\WorkSpace\C++Work\datastu\实验报告\队列-猴子分桃.exe
答案:
1 3 4 5 2 请按任意键继续. . .
```

# 实验四：字符串与数组-三元组表稀疏矩阵

代码：

```
Status TransposeSMatrix(TSMatrix M,TSMatrix &T){
    T.mu = M.nu;
    T.nu = M.mu;
    T.tu = M.tu;
    if(T.tu){
        int q = 1;
        for (int c = 1; c <= M.nu;c++){
            for (int p = 1; p <= M.tu;p++){
                if(M.data[p].j == c){
                    T.data[q].i = M.data[p].j;
                    T.data[q].j = M.data[p].i;
                    T.data[q].e = M.data[p].e;
                    q++;
                    }
            }
        }
    }
    return OK;
}

Status AddSMatrix(TSMatrix M, TSMatrix N, TSMatrix &Q){
    int m = M.tu;
    int n = N.tu;
    int c = 1,tu = 0;
    for (int i = 1,j = 1; i <= m || j <= n;){
        if(i > m){
            Q.data[c].i = N.data[j].i;
            Q.data[c].j = N.data[j].j;
            Q.data[c].e = N.data[j].e + 0;
            c++, j++, tu++;
        }
        else if(j > n){
            Q.data[c].i = M.data[i].i;
            Q.data[c].j = M.data[i].j;
            Q.data[c].e = M.data[i].e + 0;
            c++, i++, tu++;
        }
        else
        {
            if (M.data[i].i == N.data[j].i && M.data[i].j == N.data[j].j) {
                Q.data[c].i = M.data[i].i;
                Q.data[c].j = M.data[i].j;
                Q.data[c].e = M.data[i].e + N.data[j].e;
                c++, i++, j++, tu++;
            }
            else if (M.data[i].i < N.data[j].i)
            {
                Q.data[c].i = M.data[i].i;
                Q.data[c].j = M.data[i].j;
                Q.data[c].e = M.data[i].e + 0;
                c++, i++, tu++;
            }
            else if (M.data[i].i == N.data[j].i){
                if (M.data[i].j <= N.data[j].j){
                    Q.data[c].i = M.data[i].i;
                    Q.data[c].j = M.data[i].j;
                    Q.data[c].e = M.data[i].e + 0;
                    c++, i++, tu++;
                }
```

```
            else
            {
                Q.data[c].i = N.data[j].i;
                Q.data[c].j = N.data[j].j;
                Q.data[c].e = N.data[j].e + 0;
                c++, j++, tu++;
            }

        }
        else{
            Q.data[c].i = N.data[j].i;
            Q.data[c].j = N.data[j].j;
            Q.data[c].e = N.data[j].e + 0;
            c++, j++, tu++;
        }
    }
}
Q.mu = M.mu;
Q.nu = M.nu;
Q.tu = tu;
return OK;
}
```

结果：

```
J:\WorkSpace\C++Work\datastu\实验报告\字符串与数组-三元组表稀疏矩阵.exe
数组M:
1 0 0
0 2 0
0 0 3

数组N:
5 4 3
5 4 3
5 4 3

数组Q=M+N:
6 4 3
5 6 3
5 4 6

数组Q的转置:
6 5 5
4 6 4
3 3 6

请按任意键继续. . .
```

# 实验五：树-层序遍历

代码：

```
Status HierarchicalOrderTraverse(BiTree T, Status (*Visit)(ElemType e)){
    SqQueue Q;
    InitQueue(Q);
    BiTNode p;

    if(T){
        EnQueue(Q,*T);    //根节点入栈
        while(!QueueEmpty(Q)){
            DeQueue(Q,p);    //出栈
            cout<<p.data<<" ";
            if(p.lchild) {
                EnQueue(Q,*(p.lchild)); //左子树入栈
            }
            if(p.rchild) {
                EnQueue(Q,*(p.rchild)); //右子树入栈
            }


        }
    }
    return OK;
}
```

结果：

```
J:\WorkSpace\C++Work\datastu\实验报告\树-前序-中序-后序-层序遍历.exe
前序遍历二叉树(ABC  DE G  F   ):
ABC  DE G  F


前序遍历:
ABCDEGF

中序遍历:
CBEGDFA

后序遍历:
CGEFDBA

层序遍历:
A B C D E F G
--------------------------------
Process exited after 2.508 seconds with return value 0
请按任意键继续. . .
```

# 实验六：图-广度优先搜索

代码：

```
void bfs(ALgraph g,int v)
{
    ArcNode *p; //创建边
    int w;
    VNode u;
    Pseqqueue q;  //创建队列
    q=init();
    visit(g,v);  //访问节点
    visited[v]=true;
    Push(q,g,v);
    int v1;
    for(v1=0;v1<g.vertexnum;v1++)
    visited[v1]=false;
    while(!empty(q))
    {
        Pop(q,&u);  //出队
        for(p=u.firstedge;p;p=p->next)
        {
            w=p->adjvex;
            if(!visited[w])
            {
                visit(g,w);
                visited[w]=true;
                Push(q,g,w);  //节点未访问就入队
            }
        }
    }
}
```

结果：

```
J:\WorkSpace\C++Work\datastu\实验报告\图-广搜-深搜.exe
请输入图的顶点数和弧数:
8,9
请输入顶点信息:
12345678
请输入边数据:
0,1 0,2 1,3 1,4 2,5 2,6 5,6 3,7 4,7
深度优先遍历:12485367
广度优先遍历:12345678

--------------------------------
Process exited after 151 seconds with return value 0
请按任意键继续. . .
```

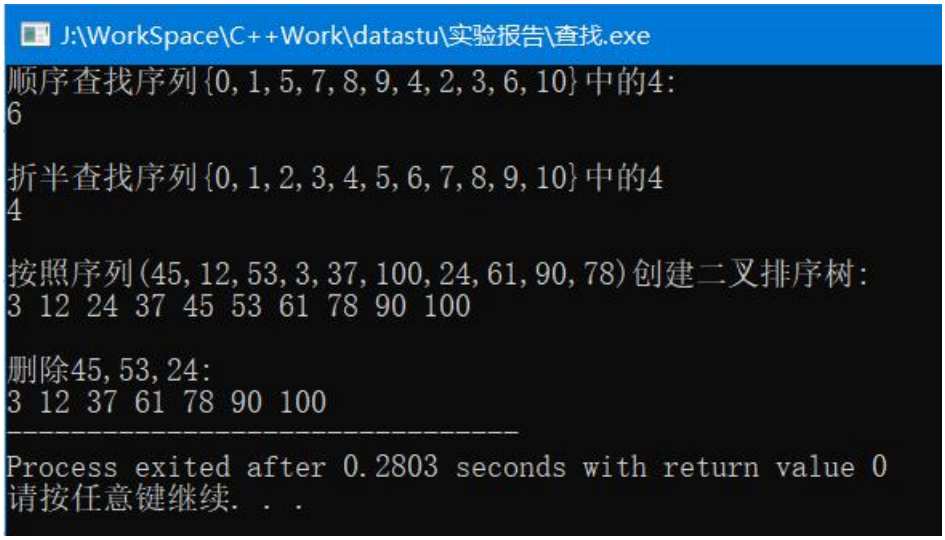
# 实验七：查找-二叉排序树

代码：

```
int InsertBST(BiTree &T,int e){
    BiTree p;
    if(!SearchBST(T,e,NULL,p)){
        BiTree s;
        s = (BiTree)malloc(sizeof(BiTNode));
        s->data = e;
        s->lchild = s->rchild = NULL;
        if(!p) T=s;
        else if(e < p->data) p->lchild = s;
        else p->rchild = s;
        return TRUE;
    }
    else return FALSE;
}
int DeleteBST(BiTree &T,int key){
    if(!T) return FALSE;
    else{
        if(T->data == key) {
        return Delete(T); //调用删除二叉树节点的函数
        }
        else if(T->data > key) return DeleteBST(T->lchild,key);
        else return DeleteBST(T->rchild,key);
    }

}
int Delete(BiTree &p){
    if(!p->rchild){
        BiTree q = p;
        p = p->lchild;
        free(q);
    }
    else if(!p->lchild){
        BiTree q = p;
        p = p->rchild;
        free(q);
    }
    else {
        BiTree q = p;
        BiTree s = p->lchild;
        while(s->rchild){
            q = s;s = s->rchild;
        }
        p->data = s->data;
        if(q!=p) q->rchild = s->lchild;
        else q->lchild = s->lchild;
        delete s;
    }
    return TRUE;
}
```

结果：


J:\WorkSpace\C++Work\datastu\实验报告\查找.exe
顺序查找序列{0,1,5,7,8,9,4,2,3,6,10}中的4:
6
折半查找序列{0,1,2,3,4,5,6,7,8,9,10}中的4
4
按照序列(45,12,53,3,37,100,24,61,90,78)创建二叉排序树:
3 12 24 37 45 53 61 78 90 100
删除45,53,24:
3 12 37 61 78 90 100
Process exited after 0.2803 seconds with return value 0
请按任意键继续. . .

# 实验八：排序-快速排序

代码：

```
int Partition(SqList &L,int low,int high){
    int pivo = L.elem[low];
    while(low<high){
        //从右往左搜索，遇到比pivo小的数就与low交换
        while(low<high&&L.elem[high]>=pivo) high--;
        int z = L.elem[low];
        L.elem[low] = L.elem[high];
        L.elem[high] = z;

        //从左往右搜索，遇到比pivo大的数就与high交换
        while(low<high&&L.elem[low]<=pivo) low++;
        z = L.elem[low];
        L.elem[low] = L.elem[high];
        L.elem[high] = z;

    }
    return low; //返回pivo最后的位置
}

void QSort(SqList &L,int low,int high){
    if(low < high){
        int pivotloc = Partition(L,low,high);
        QSort(L,low,pivotloc-1); //对左边序列进行快排
        QSort(L,pivotloc+1,high); //对右边序列进行快排
    }
}
```

结果：

```
J:\WorkSpace\C++Work\datastu\实验报告\排序-插入-折半插入-快排-选择.exe
插入排序:
49 38 65 97 76 13 27
13 27 38 49 65 76 97

折半插入排序:
49 38 65 97 76 13 27
13 27 38 49 65 76 97

快速排序:
49 38 65 97 76 13 27
13 27 38 49 65 76 97

选择排序:
49 38 65 97 76 13 27
13 27 38 49 65 76 97

--------------------------------
Process exited after 0.2773 seconds with return value 0
请按任意键继续. . .
```