20220402-数据结构

1.过程描述

顺序栈的表示和实现 C++ 复制代码

```
顺序栈的存储结构
#define MAXSIZE 100
typedef struct
{
    SElemType *base;
    SElemType *top;
    int stacksize;
}SqStack;

1.初始化
Status InitStack(SqStack &S)
{
    S.base=new SElemType[MAXSIZE];
    if(!S.base)
        exit(OVERFLOW);//存储分配失败
    S.top=S.base;
    S.stacksize=MAXSIZE;
    return OK;
}

2.入栈
Status Push(SqStack &S,SElemType e)
{
    if(S.top-S.base==S.stacksize)
        return ERROR;
    *S.top++=e;//元素e压入栈顶，栈顶指针加1
    //*S.top=e;
    //S.top++;
    return OK;
}

3.出栈
Status Pop(SqlStack &S,SElemType &e)
{
    if(S.top==S.base)
        return ERROR;//栈空
    e=*--S.top;//栈顶指针减一，将栈顶元素赋给e
    return OK;
}

4.取栈顶元素
SElemType GetTop(SqStack S)
{
    if(S.top!=S.base)
        return *(S.top-1);//返回栈顶元素的值，栈顶指针不变
```

```
46    }
```

链栈的表示和实现 C++ 复制代码

```cpp
typedef struct StackNode
{
    ElemType data;
    struct StackNode *next;
}StackNode,*LinkStack;

1.初始化
Status InitStack(LinkStack &S)
{
    S=nullptr;
    return OK;
}

2.入栈
Status Push(LinkStack &S,SElemType e)
{
    p=new StackNode;
    p->data=e;
    p->next=S;
    S=p;
    return OK;
}

3.出栈
Status Pop(LinkList &S,SElemType e)
{
    if(S==nullptr)
        return ERROR;
    e=S->data;
    p=S;
    S=p->next;
    delete S;
    return OK;
}

4.取栈顶元素
SELemType GetTop(LinkStack S)
{
    if(S!=nullptr)
        S->data;
}
```

递归 C++ 复制代码

```cpp
1.遍历链表中各个节点的递归算法
void TraverseList(LinkList p)
{
    if(p)
    {
        cout<<p->data<<endl;
        TraverseList(p->next);
    }
}

2.Hanoi塔问题(与斐波那契数列一样，时间复杂度为O(2^n)
int m=0;
void move(char A,int n,char C)
{
    cout<<++m<<","<<n<<","<<A<<","<<C<<endl;
}

void Hanoi(int n,char A,char B,char C)
{
    if(n==1)
        move(A,1,C);
    else
    {
        Hanoi(n-1,A,C,B);
        move(A,n,C);
        Hanoi(n-1,B,A,C);
    }
}

```

队列的顺序实现和表示 C++ 复制代码

```
队列的顺序存储结构
#define MAXSIZE 100
typedef struct
{
    QElemType *base//存储空间的基地址
    int front;//头指针
    int rear;//尾指针
}SqQueue;

1.初始化
Status InitQueue(SqQueue &Q)
{
    Q.base=new QElemType[MAXSIZE];//分配一个最大容量的数组空间
    if(!Q.base)
        exit(OVERFLOW);//存储分配失败
    Q.front=Q.rear=0;//头指针和尾指针设置为零，队列为空
    return OK;
}

2.求队列长度
int QueueLength(SqQueue Q)
{
    return(Q.rear-Q.front+MAXSIZE)%MAXSIZE;
}

3.入队
Status EnQueue(SqQueue &Q,QElemType e)
{
    if(Q.rear+1)%MAXSIZE==Q.front//尾指针在循环意义上加1后等于头指针，表明队满
        return ERROR;
    Q.base[Q.rear]=e;//新元素插入队尾
    Q.rear=(Q.rear+1)%MAXSIZE;//队尾指针加1
    return OK;
}

4.出队
Status DeQueue(SqQueue &Q,QElemType &e)
{
    if(Q.front==Q.rear)
        return ERROR;
    e=Q.base[Q.front];//保存队头元素
    Q.front=(Q.front+1)%MAXSZIE;//对头指针加1
    return OK;
}

```

```
5.取队头元素
SElemType GetHead(SqQueue Q)
{
    if(Q.front!=Q.rear)
        return Q.base[Q.front];
}
```

队列的链式表示和实现 C++ 复制代码

```
队列的链式存储结构
typedef struct QNode
{
    QElemType data;
    struct QNode *next;
}QNode,*QueuePtr;

typedef struct
{
    QueuePtr front;
    QueuePtr rear;
}LinkQueue;

1.初始化
Status InitQueue(LinkQueue &Q)
{
    Q.front=Q.rear=new QNode;
    Q.front->next=nullptr;
    return OK;
}

2.入队
Status EnQueue(LinkQueue &Q,QElemType e)
{
    p=new QNode;
    p->data=e;
    p->next=nullptr;
    Q.rear->next=p;
    Q.rear=p;
    return OK;
}

3.出队
Status DeQueue(LinkQueue &Q,QElemType &e)
{
    if(Q.front==Q.rear)
        return ERROR;
    p=Q.front->next;
    e=p->data;
    Q.front->next=p->next;
    if(Q.rear==p)
        Q.rear=Q.front;
    delete p;
    return OK;
}
```

```
46
47  4.取队头元素
48  SElemType GetHead(LinkQueue Q)
49  {
50      if(Q.front!=Q.rear)
51          return Q.front->next->data;
52  }
```

应用案例 C++ 复制代码

```
1.数制转换
void conversion(int N)
{
    InitStack(S);
    while(N)
    {
        Push(S,N%8);
        N=N/8;
    }
    while(!StackEmpty(S))
    {
        Pop(S,e);
        cout<<e;
    }
}
该算法的时间和空间复杂度为O(log8n)

2.括号匹配的检验
Status Matching()
{
    InitStack(S);
    flag=1;
    cin>>ch;
    while(ch!='#'&&flag)
    {
        switch(ch)
        {
            case '{' || '(':
                Push(S,ch);
                break;
            case ')':
                if(!StackEmpty(S) && GetTop(S)=='(')
                    Pop(S,x);
                else
                    flag=0;
                break;
            case ']':
                if(!StackEmpty(S)&&GetTop(S)=='[')
                    Pop(S,x);
                else
                    flag=0;
                break;
        }
        cin>>ch;
    }
```

```
46          if(StackEmpty(S)&&flag)
47              return true;
48      }
49
```

2.结果输出

今天粗略地看了栈和队列部分，可能还是基础太薄弱，感觉教材看着越来越吃力，很多伪代码看的摸不着头脑，有点跟不上作者的节奏，后来上豆瓣查了下，发现教材得分不高，底下也一堆人在吐槽，决定及时止损，换本口碑比较好的看。这两天基本算白费了，加上最近学习状态不怎么好，隐隐又察觉到拖延症犯病的征兆。

这两天注意力不集中的时候也查了查关于计算机的一些资料，发现确实浩如烟海，而且十分注重实践，想在几个月内就达到什么什么水平真的是不现实，只能是先奠定一个持续学习的基础。看着这么多东西，什么都想学，但现阶段学起来确实比较费劲，急也急不得，基础不牢后面就更加举步维艰了。收收心，脚踏实地。