栈

1.顺序栈

```
#include <stdio.h>
#include <malloc.h>
#define MaxSize 100

typedef char ElemType;

typedef struct
{
    ElemType data[MaxSize];  //存放栈中的数据元素
    int top;                 //栈顶指针，即存放栈顶元素在data数组中的下标
} SqStack;                   //顺序栈类型

void InitStack(SqStack *&s) //初始化栈
{
    s=(SqStack *)malloc(sizeof(SqStack)); //分配一个顺序栈空间，首地址存放在s中
    s->top=-1; //栈顶指针置为-1
}

void DestroyStack(SqStack *&s) //销毁栈
{
    free(s); //释放指针
}

bool StackEmpty(SqStack *s) //判断栈是否为空
{
    return(s->top==-1);
}

bool Push(SqStack *&s,ElemType e) //进栈
{
    if (s->top==MaxSize-1)    //栈满的情况，即栈上溢出
        return false;
    s->top++; //栈顶指针增加1
    s->data[s->top]=e; //元素e放在栈顶指针处
    return true;
}

bool Pop(SqStack *&s,ElemType &e) //出栈
{
    if (s->top==-1)     //栈为空的情况，即栈下溢出
        return false;
    e=s->data[s->top]; //取栈顶元素
    s->top--; //栈顶指针减1
    return true;
}

bool GetTop(SqStack *s,ElemType &e) //取栈顶元素
{
```

```
    if (s->top==-1)          //栈为空的情况，即栈下溢出
        return false;
    e=s->data[s->top]; //取栈顶元素
    return true;
}
```

2.链栈

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

typedef char ElemType;

typedef struct linknode
{
    ElemType data;              //数据域
    struct linknode *next;      //指针域
} LinkStNode;                   //链栈结点类型

void InitStack(LinkStNode *&s) //初始化栈
{
    s=(LinkStNode *)malloc(sizeof(LinkStNode));
    s->next=NULL;
}

void DestroyStack(LinkStNode *&s) //销毁栈
{
    LinkStNode *pre=s,*p=s->next; //pre指向头结点，p指向首结点
    while (p!=NULL) //循环到p为空
    {
        free(pre); //释放pre结点
        pre=p; //pre p同步后移
        p=p->next;
    }
    free(pre);  //pre指向尾结点,释放其空间
}

bool StackEmpty(LinkStNode *s) //判断栈是否为空
{
    return(s->next==NULL);
}

void Push(LinkStNode *&s,ElemType e) //进栈
{   LinkStNode *p;
    p=(LinkStNode *)malloc(sizeof(LinkStNode)); //新建元素e对应的结点p
    p->data=e;              //存放元素e
    p->next=s->next;        //插入p结点作为首结点
    s->next=p;
}

bool Pop(LinkStNode *&s,ElemType &e) //出栈
{   LinkStNode *p;
    if (s->next==NULL)      //栈空的情况
        return false;
    p=s->next;              //p指向首结点
```

```
    e=p->data;                  //提取首结点
    s->next=p->next;            //删除首结点，建立新链
    free(p);                    //释放被删结点的存储空间
    return true;
}

bool GetTop(LinkStNode *s,ElemType &e) //取栈顶元素
{   if (s->next==NULL)          //栈空的情况
        return false;
    e=s->next->data;            //提取首结点值
    return true;
}
```

队列

1.顺序队列

```
//非环形顺序队列
#include <stdio.h>
#include <malloc.h>
#define MaxSize 100

typedef char ElemType;

typedef struct
{
    ElemType data[MaxSize];
    int front,rear;                         //队头和队尾指针
} SqQueue;

void InitQueue(SqQueue *&q)
{   q=(SqQueue *)malloc (sizeof(SqQueue));
    q->front=q->rear=-1;
}

void DestroyQueue(SqQueue *&q)              //销毁队列
{
    free(q);
}

bool QueueEmpty(SqQueue *q)                 //判断队列是否为空
{
    return(q->front==q->rear);
}

bool enQueue(SqQueue *&q,ElemType e)        //进队
{   if (q->rear==MaxSize-1)                 //队满上溢出
        return false;                       //返回假
    q->rear++;                              //队尾增1
    q->data[q->rear]=e;                     //rear位置插入元素e
    return true;                            //返回真
}

bool deQueue(SqQueue *&q,ElemType &e)       //出队
{   if (q->front==q->rear)                  //队空下溢出
```

```
        return false;
    q->front++;
    e=q->data[q->front];
    return true;
}

//环形顺序队列

#include <stdio.h>
#include <malloc.h>
#define MaxSize 100

typedef char ElemType;

typedef struct
{
    ElemType data[MaxSize]; //存放队中元素
    int front,rear;     //队首和队尾指针
} SqQueue; //顺序队类型

void InitQueue(SqQueue *&q) //初始化队列
{   q=(SqQueue *)malloc (sizeof(SqQueue));
    q->front=q->rear=0;
}

void DestroyQueue(SqQueue *&q) //销毁队列
{
    free(q);
}

bool QueueEmpty(SqQueue *q) //判断队列是否为空
{
    return(q->front==q->rear);
}

bool enQueue(SqQueue *&q,ElemType e) //环形队列进队列
{   if ((q->rear+1)%MaxSize==q->front)  //队满真溢出
        return false;
    q->rear=(q->rear+1)%MaxSize;
    q->data[q->rear]=e;
    return true;
}

bool deQueue(SqQueue *&q,ElemType &e) //环形队列出队列
{   if (q->front==q->rear)      //队空下溢出
        return false;
    q->front=(q->front+1)%MaxSize;
    e=q->data[q->front];
    return true;
}
```

2.链队

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

typedef char ElemType;

typedef struct DataNode
{
    ElemType data; //存放元素
    struct DataNode *next; //下一个结点指针
} DataNode;             //链队数据结点类型

typedef struct
{
    DataNode *front; //指向队首结点
    DataNode *rear; //指向队尾结点
} LinkQuNode;           //链队结点的类型

void InitQueue(LinkQuNode *&q) //初始化队列
{
    q=(LinkQuNode *)malloc(sizeof(LinkQuNode));
    q->front=q->rear=NULL;
}

void DestroyQueue(LinkQuNode *&q) //销毁队列
{
    DataNode *pre=q->front,*p;//pre指向队首结点
    if (pre!=NULL)
    {   p=pre->next; //p指向结点pre的后继节点
        while (p!=NULL) //p不空时循环
        {   free(pre); //释放pre结点
            pre=p;p=pre->next; //pre 和 p 同步后移
        }
    }
    free(pre); //释放最后一个数据节点
    free(q); //释放链队结点占用空间
}

bool QueueEmpty(LinkQuNode *q) //判断队列是否为空
{
    return(q->rear==NULL);
}

void enQueue(LinkQuNode *&q,ElemType e)
{   DataNode *p;
    p=(DataNode *)malloc(sizeof(DataNode)); //创建新结点
    p->data=e;
    p->next=NULL;
    if (q->rear==NULL)      //若链队为空,则新结点是队首结点又是队尾结点
        q->front=q->rear=p;
    else                    //若队列不空
    {   q->rear->next=p;    //将p结点链到队尾,并将rear指向它
        q->rear=p;
    }
```

```
}

bool deQueue(LinkQuNode *&q,ElemType &e) //出队列
{   DataNode *t;
    if (q->rear==NULL)      //队列为空
        return false;
    t=q->front;             //t指向首结点
    if (q->front==q->rear)  //队列中只有一个数据结点时
        q->front=q->rear=NULL;
    else                    //队列中有多个数据结点时
        q->front=q->front->next;
    e=t->data;
    free(t);
    return true;
}
```