第2章 栈和队列

2.1 栈

2.1.1 顺序栈

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAXSIZE 100                    /*  顺序栈空间的存储量  */
typedef int DataType;                  /*  顺序栈元素类型  */
typedef struct{
    DataType *data;                    /*  顺序栈元素存储空间基址  */
    int top;                           /*  栈顶  */
}SeqStack;                             /*  顺序栈类型  */
void wait( )
{
    printf("\n 请按任意键...\n");
    getch( );
}
int go_on( )
{
    int flag=1;
    char choice;
    while(1){
        printf("\n 继续吗？[Y/N]");
        choice=getche( );
        if(choice=='Y' || choice=='y')
            break;
        else if(choice=='N' || choice=='n'){
            flag=0;
            break;
        }
```

```
    }
    return(flag);
}
/* 初始化，构造一个空顺序栈 */
void Init_SeqStack(SeqStack *S,int n)
{
    S->data=(DataType *)malloc(n*sizeof(DataType));
    if(S->data==NULL){
        printf("\n 内存分配失败.\n");
        exit(1);
    }
    S->top=-1;
}
/* 判断顺序栈是否为空 */
int Empty_SeqStack(SeqStack *S)
{
    if(S->top==-1)
        return(1);
    else
        return(0);
}
/* 判断顺序栈是否为满 */
int Full_SeqStack(SeqStack *S)
{
    if(S->top==MAXSIZE-1)
        return(1);
    else
        return(0);
}
/* 入栈，插入元素 e 为新的栈顶元素 */
int Push_SeqStack(SeqStack *S,DataType e)
{
    if(Full_SeqStack(S)==1){        /* 栈满，不能入栈 */
        printf("\n 栈满，不能入栈.\n");
        return(0);
```

```
    }
    else{
        S->top++;
        S->data[S->top]=e;              /* 元素 e 入栈 */
        return(1);
    }
}
void Push(SeqStack *S)
{
    DataType x;
    int flag=1,push_flag;
    while(flag){
        printf("\n 请输入要入栈元素:");
        scanf("%d",&x);
        push_flag=Push_SeqStack(S,x);
        if(push_flag==1)
            printf("\n 入栈成功.\n");
        else
            printf("\n 入栈失败.\n");
        flag=go_on( );
    }
}
/* 出栈，删除栈顶元素，并由*e 返回其值 */
int Pop_SeqStack(SeqStack *S,DataType *e)
{
    if(Empty_SeqStack(S)){              /* 栈空，不能出栈 */
        printf("\n 栈空，不能出栈.\n");
        return(0);
    }
    else{
        *e=S->data[S->top];             /* 栈顶元素存入*e */
        S->top--;
        return(1);
    }
}
```

```
void Pop(SeqStack *S)
{
    DataType x;
    int flag=1,pop_flag;
    while(flag){
        pop_flag=Pop_SeqStack(S,&x);
        if(pop_flag==1)
            printf("\n 出栈成功，出栈元素为：%d\n",x);
        else
            printf("\n 出栈失败.\n");
        flag=go_on( );
    }
}
/* 取栈顶元素，由*e 返回其值 */
int GetTop_SeqStack(SeqStack *S,DataType *e)
{
    if(Empty_SeqStack(S)){              /* 栈空，不能取栈顶元素 */
        printf("\n 栈空，不能取栈顶元素.\n");
        return(0);
    }
    else{
        *e=S->data[S->top];
        return(1);
    }
}
/* 输出栈顶元素 */
void Display_Top(SeqStack *S)
{
    DataType e;
    if(Empty_SeqStack(S)==1)
        printf("\n 栈空，没有元素.\n");
    else{
        GetTop_SeqStack(S,&e);
        printf("\n 栈顶元素： ");
        printf("%4d\n",e);
```

```
    }
}
/* 输出栈全部元素 */
void Display_SeqStack(SeqStack *S)
{
    int i;
    if(Empty_SeqStack(S)==1)
        printf("\n 栈空，没有元素.\n");
    else{
        printf("\n 栈全部元素");
        printf("\n 栈底<---------------->栈顶\n");
        for(i=0;i<=S->top;i++)
            printf("%6d",S->data[i]);
        printf("\n");
    }
}
main( )
{
    SeqStack S;
    char choice;
    int flag=1;
    Init_SeqStack(&S,MAXSIZE);
    do{
        printf("\n");
        printf("----顺序栈（动态数组实现）----\n");
        printf("      1.......入栈\n");
        printf("      2.......出栈\n");
        printf("      3.......输出栈顶元素\n");
        printf("      4.......输出全部元素\n");
        printf("      0.......退出\n");
        printf("------------------------------\n");
        printf("请选择[1/2/3/4/0]:");
        choice=getche( );
        switch(choice){
            case '1':Push(&S);break;
```

```
            case '2':Pop(&S);break;
            case '3':Display_Top(&S);break;
            case '4':Display_SeqStack(&S);break;
            case '0':flag=0;break;
        }
        wait( );
    }while(flag==1);
}
```

2.1.2 链栈

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
typedef int DataType;                   /* 链栈元素类型 */
typedef struct node{
    DataType data;                      /* 链栈元素 */
    struct node *next;                  /* 链栈元素后继指针 */
}SNode,*LinkStack;                      /* 链栈结点类型、链栈类型 */
void wait( )
{
    printf("\n 请按任意键...\n");
    getch( );
}
int go_on( )
{
    int flag=1;
    char choice;
    while(1){
        printf("\n 继续吗？[Y/N]");
        choice=getche( );
        if(choice=='Y' || choice=='y')
            break;
        else if(choice=='N' || choice=='n'){
            flag=0;
```

```
                break;
            }
        }
        return(flag);
}
/* 初始化，构造一个空的带头结点链栈 */
void Init_LinkStack(LinkStack *S)
{
        *S=(SNode *)malloc(sizeof(SNode));
        if(*S==NULL){
            printf("\n 内存分配失败.\n");
            exit(-1);
        }
        (*S)->next=NULL;
}
/* 判断带头结点链栈是否为空 */
int Empty_LinkStack(LinkStack S)
{
        if(S->next==NULL)
            return(1);
        else
            return(0);
}
/* 入栈，插入元素 e 为新的栈顶元素 */
int Push_LinkStack(LinkStack S,DataType e)
{
        SNode *q;
        q=(SNode *)malloc(sizeof(SNode));
        if(q==NULL){
            printf("\n 内存分配失败.\n");
            return(0);
        }
        q->data=e;
        q->next=S->next;
        S->next=q;
```

```
    return(1);
}
void Push(LinkStack S)
{
    DataType x;
    int flag=1,push_flag;
    while(flag){
        printf("\n 请输入要入栈元素:");
        scanf("%d",&x);
        push_flag=Push_LinkStack(S,x);
        if(push_flag==1)
            printf("\n 入栈成功.\n");
        else
            printf("\n 入栈失败.\n");
        flag=go_on( );
    }
}
/* 出栈，删除栈顶元素，并由*e 返回其值 */
int Pop_LinkStack(LinkStack S,DataType *e)
{
    SNode *p=S->next;
    if(Empty_LinkStack(S)){          /* 栈空，不能出栈 */
        printf("\n 栈空，不能出栈.\n");
        return(0);
    }
    else{
        *e=p->data;                  /* 栈顶元素存入*e */
        S->next=p->next;
        free(p);
        return(1);
    }
}
void Pop(LinkStack S)
{
    DataType x;
```

```
    int flag=1,pop_flag;
    while(flag){
        pop_flag=Pop_LinkStack(S,&x);
        if(pop_flag==1)
            printf("\n 出栈成功，出栈元素为：%d\n",x);
        else
            printf("\n 出栈失败.\n");
        flag=go_on( );
    }
}
/* 取栈顶元素，由*e 返回其值 */
int GetTop_LinkStack(LinkStack S,DataType *e)
{
    if(Empty_LinkStack(S)){          /* 栈空，不能取栈顶元素 */
        printf("\n 栈空，不能取栈顶元素.\n");
        return(0);
    }
    else{
        *e=S->next->data;
        return(1);
    }
}
/* 输出栈顶元素 */
void Display_Top(LinkStack S)
{
    DataType e;
    if(Empty_LinkStack(S)==1)
        printf("\n 栈空，没有元素.\n");
    else{
        GetTop_LinkStack(S,&e);
        printf("\n 栈顶元素： ");
        printf("%4d\n",e);
    }
}
/* 输出栈全部元素 */
```

```
void Display_LinkStack(LinkStack S)
{
    SNode *p=S->next;
    if(Empty_LinkStack(S)==1)
        printf("\n 栈空，没有元素.\n");
    else{
        printf("\n 栈全部元素");
        printf("\n 栈顶<---------------->栈底\n");
        while(p!=NULL){
            printf("%4d",p->data);
            p=p->next;
        }
    }
}
main( )
{
    LinkStack S;
    char choice;
    int flag=1;
    Init_LinkStack(&S);
    do{
        printf("\n");
        printf("-------链栈（带头结点）-------\n");
        printf("      1.......入栈\n");
        printf("      2.......出栈\n");
        printf("      3.......输出栈顶元素\n");
        printf("      4.......输出全部元素\n");
        printf("      0.......退出\n");
        printf("------------------------------\n");
        printf("请选择[1/2/3/4/0]:");
        choice=getche( );
        switch(choice){
            case '1':Push(S);break;
            case '2':Pop(S);break;
            case '3':Display_Top(S);break;
```

```
            case '4':Display_LinkStack(S);break;
            case '0':flag=0;break;
        }
        wait( );
    }while(flag==1);
}
```

2.2 队列

2.2.1 环状队列

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
#define MAXSIZE 100                 /* 环状队列空间的存储量 */
typedef int DataType;               /* 环状队列元素类型 */
typedef struct{
    DataType *data;                 /* 环状队列元素存储空间基址 */
    int front,rear;                 /* 环状队列头、尾 */
}SeqQueue;                          /* 环状队列类型 */
void wait( )
{
    printf("\n 请按任意键...\n");
    getch( );
}
int go_on( )
{
    int flag=1;
    char choice;
    while(1){
        printf("\n 继续吗？[Y/N]");
        choice=getche( );
        if(choice=='Y' || choice=='y')
            break;
        else if(choice=='N' || choice=='n'){
```

```
                flag=0;
                break;
            }
        }
        return(flag);
}
/* 初始化，构造一个空环状队列 */
void Init_SeqQueue(SeqQueue *Q,int n)
{
        Q->data=(DataType *)malloc(n*sizeof(DataType));
        if(Q->data==NULL){
            printf("\n 内存分配失败.\n");
            exit(-1);
        }
        Q->front=Q->rear=0;
}
/* 判断环状队列是否为空 */
int Empty_SeqQueue(SeqQueue *Q)
{
        if(Q->front==Q->rear)
            return(1);
        else
            return(0);
}
/* 判断环状队列是否为满 */
int Full_SeqQueue(SeqQueue *Q)
{
        if((Q->rear+1)%MAXSIZE==Q->front)
            return(1);
        else
            return(0);
}
/* 求环状队列队长 */
int Length_SeqQueue(SeqQueue *Q)
{
```

```
    int k;
    k=(Q->rear-Q->front+MAXSIZE)%MAXSIZE;
    return(k);
}
void Length(SeqQueue *Q)
{
    int k;
    k=Length_SeqQueue(Q);
    printf("\n 队长：%d\n",k);
}
/* 入队，插入元素 e 为新的队头元素 */
int EnQueue_SeqQueue(SeqQueue *Q,DataType e)
{
    if(Full_SeqQueue(Q)==1){          /* 队满，不能入队 */
        printf("队满，不能入队.\n");
        return(0);
    }
    else{
        Q->data[Q->rear]=e;           /* 元素 e 入队 */
        Q->rear=(Q->rear+1)%MAXSIZE;
        return(1);
    }
}
void EnQueue(SeqQueue *Q)
{
    DataType x;
    int flag=1,enqueue_flag;
    while(flag){
        printf("\n 请输入要入队元素:");
        scanf("%d",&x);
        enqueue_flag=EnQueue_SeqQueue(Q,x);
        if(enqueue_flag==1)
            printf("\n 入队成功.\n");
        else
            printf("\n 入队失败.\n");
```

```
            flag=go_on( );
        }
    }
    /* 出队，删除队头元素，并由*e 返回其值 */
    int DeQueue_SeqQueue(SeqQueue *Q,DataType *e)
    {
        if(Empty_SeqQueue(Q)){          /* 队空，不能出队 */
            printf("\n 队空，不能出队.\n");
            return(0);
        }
        else{
            *e=Q->data[Q->front];       /* 队头元素存入*e */
            Q->front=(Q->front+1)%MAXSIZE;
            return(1);
        }
    }
    void DeQueue(SeqQueue *Q)
    {
        DataType x;
        int flag=1,dequeue_flag;
        while(flag){
            dequeue_flag=DeQueue_SeqQueue(Q,&x);
            if(dequeue_flag==1)
                printf("\n 出队成功，出队元素为：%d\n",x);
            else
                printf("\n 出队失败.\n");
            flag=go_on( );
        }
    }
    /* 取队头元素，由*e 返回其值 */
    int GetFront_SeqQueue(SeqQueue *Q,DataType *e)
    {
        if(Empty_SeqQueue(Q)){          /*队空，不能取队头元素*/
            printf("\n 队空，不能取队头元素.\n");
            return(0);
```

```
    }
    else{
        *e=Q->data[Q->front];
        return(1);
    }
}
/* 输出队头元素 */
void Display_Front(SeqQueue *Q)
{
    DataType e;
    if(Empty_SeqQueue(Q)==1)
        printf("\n 队空，没有元素.\n");
    else{
        GetFront_SeqQueue(Q,&e);
        printf("\n 队头元素：\n");
        printf("%4d\n",e);
    }
}
/* 输出队全部元素 */
void Display_SeqQueue(SeqQueue *Q)
{
    int k,len;
    if(Empty_SeqQueue(Q)==1)
        printf("\n 队空，没有元素.\n");
    else{
        printf("\n 队全部元素\n");
        printf("\n 队头<---------->队尾\n");
        len=Length_SeqQueue(Q);
        for(k=0;k<len;k++)
            printf("%4d",Q->data[(Q->front+k)%MAXSIZE]);
    }
}
main( )
{
    SeqQueue Q;
```

```
    char choice;
    int flag=1;
    Init_SeqQueue(&Q,MAXSIZE);
    do{
        printf("\n");
        printf("----环状顺序队列（动态数组实现）----\n");
        printf("          1.......入队元素\n");
        printf("          2.......出队元素\n");
        printf("          3.......输出队长\n");
        printf("          4.......输出队头元素\n");
        printf("          5.......输出全部元素\n");
        printf("          0.......退出\n");
        printf("-----------------------------------\n");
        printf("请选择[1/2/3/4/5/0]:");
        choice=getche( );
        switch(choice){
            case '1':EnQueue(&Q);break;
            case '2':DeQueue(&Q);break;
            case '3':Length(&Q);break;
            case '4':Display_Front(&Q);break;
            case '5':Display_SeqQueue(&Q);break;
            case '0':flag=0;break;
        }
        wait( );
    }while(flag==1);
}
```

2.2.2 链队

```
#include <stdio.h>
#include <stdlib.h>
#include <conio.h>
typedef int DataType;                    /* 链队元素类型 */
typedef struct node{
    DataType data;                       /* 链队元素 */
```

```
    struct node *next;                  /* 链队元素后继指针 */
}QNode;                                 /* 链队结点类型 */
typedef struct{
    QNode *front;                       /* 链队头指针 */
    QNode *rear;                        /* 链队尾指针 */
}LinkQueue;                             /* 链队类型 */
void wait( )
{
    printf("\n 请按任意键...\n");
    getch( );
}
int go_on( )
{
    int flag=1;
    char choice;
    while(1){
        printf("\n 继续吗？[Y/N]");
        choice=getche( );
        if(choice=='Y' || choice=='y')
            break;
        else if(choice=='N' || choice=='n'){
            flag=0;
            break;
        }
    }
    return(flag);
}
/* 初始化，构造一个空的带头结点链队 */
void Init_LinkQueue(LinkQueue *Q)
{
    QNode *p;
    p=(QNode *)malloc(sizeof(QNode));
    if(p==NULL){
        printf("\n 内存分配失败.\n");
        exit(-1);
```

```
    }
    p->next=NULL;
    Q->front=Q->rear=p;
}
/* 判断带头结点链队是否为空 */
int Empty_LinkQueue(LinkQueue *Q)
{
    if(Q->front==Q->rear)
        return(1);
    else
        return(0);
}
/* 求带头结点链队长度 */
int Length_LinkQueue(LinkQueue *Q)
{
    QNode *p=Q->front->next;
    int k;
    k=0;
    while(p!=NULL){
        k++;
        p=p->next;
    }
    return(k);
}
void Length(LinkQueue *Q)
{
    int k;
    k=Length_LinkQueue(Q);
    printf("\n 队长：%d\n",k);
}
/* 入队，插入元素 e 为新的队尾元素 */
int EnQueue_LinkQueue(LinkQueue *Q,DataType e)
{
    QNode *p;
    p=(QNode *)malloc(sizeof(QNode));
```

```
    if(p==NULL){
        printf("\n 内存分配失败.\n");
        return(0);
    }
    p->data=e;
    p->next=NULL;
    Q->rear->next=p;
    Q->rear=p;
    return(1);
}
void EnQueue(LinkQueue *Q)
{
    DataType x;
    int flag=1,enqueue_flag;
    while(flag){
        printf("\n 请输入要入队元素:");
        scanf("%d",&x);
        enqueue_flag=EnQueue_LinkQueue(Q,x);
        if(enqueue_flag==1)
            printf("\n 入队成功.\n");
        else
            printf("\n 入队失败.\n");
        flag=go_on( );
    }
}
/* 出队，删除队头元素，并由*e 返回其值 */
int DeQueue_LinkQueue(LinkQueue *Q,DataType *e)
{
    QNode *p;
    if(Empty_LinkQueue(Q)){          /* 队空，不能出队 */
        printf("\n 队空，不能出队.\n");
        return(0);
    }
    else{
        p=Q->front->next;
```

```
        *e=p->data;                    /* 队头元素存入*e */
        Q->front->next=p->next;
        if(Q->rear==p)
            Q->rear=Q->front;
        free(p);
        return(1);
    }
}
void DeQueue(LinkQueue *Q)
{
    DataType x;
    int flag=1,dequeue_flag;
    while(flag){
        dequeue_flag=DeQueue_LinkQueue(Q,&x);
        if(dequeue_flag==1)
            printf("\n 出队成功，出队元素为：%d\n",x);
        else
            printf("\n 出队失败.\n");
        flag=go_on( );
    }
}
/* 取队头元素，由*e 返回其值 */
int GetFront_LinkQueue(LinkQueue *Q,DataType *e)
{
    QNode *p;
    if(Empty_LinkQueue(Q)){          /* 队空，不能取队头元素 */
        printf("\n 队空，不能取队头元素.\n");
        return(0);
    }
    else{
        p=Q->front->next;
        *e=p->data;
        return(1);
    }
}
```

```
/* 输出队头元素 */
void Display_Front(LinkQueue *Q)
{
    DataType e;
    if(Empty_LinkQueue(Q)==1)
        printf("\n 队空，没有元素.\n");
    else{
        GetFront_LinkQueue(Q,&e);
        printf("\n 队头元素：");
        printf("%4d\n",e);
    }
}
/* 输出队全部元素 */
void Display_LinkQueue(LinkQueue *Q)
{
    QNode *p=Q->front->next;
    if(Empty_LinkQueue(Q)==1)
        printf("\n 队空，没有元素.\n");
    else{
        printf("\n 队全部元素\n");
        printf("\n 队头<---------->队尾\n");
        while(p!=NULL){
            printf("%4d",p->data);
            p=p->next;
        }
    }
}
main( )
{
    LinkQueue Q;
    char choice;
    int flag=1;
    Init_LinkQueue(&Q);
    do{
        printf("\n");
```

```
        printf("----------链队（带头结点）----------\n");
        printf("          1.......入队元素\n");
        printf("          2.......出队元素\n");
        printf("          3.......输出队长\n");
        printf("          4.......输出队头元素\n");
        printf("          5.......输出全部元素\n");
        printf("          0.......退出\n");
        printf("-----------------------------------\n");
        printf("请选择[1/2/3/4/5/0]:");
        choice=getche( );
        switch(choice){
            case '1':EnQueue(&Q);break;
            case '2':DeQueue(&Q);break;
            case '3':Length(&Q);break;
            case '4':Display_Front(&Q);break;
            case '5':Display_LinkQueue(&Q);break;
            case '0':flag=0;break;
        }
        wait( );
    }while(flag==1);
}
```