数据结构实验报告——实验六

学号：_20201050331___姓名：__黄珀芝___得分：_____

一、实验目的

1、复习队列的逻辑结构、存储结构以及基本操作
2、掌握链队列、循环队列

二、实验内容

1、(必做题)假设队列中数据元素类型是字符型,请采用
链队列实现队列的以下基本操作
(1) Status InitQucue（&q)∥构造空队列 Q
(2) Status EnQueue(&q,e)∥元素 e 入队列 Q;
(3)Status DeQucue(&q,&e)队列出队列,元素为 e

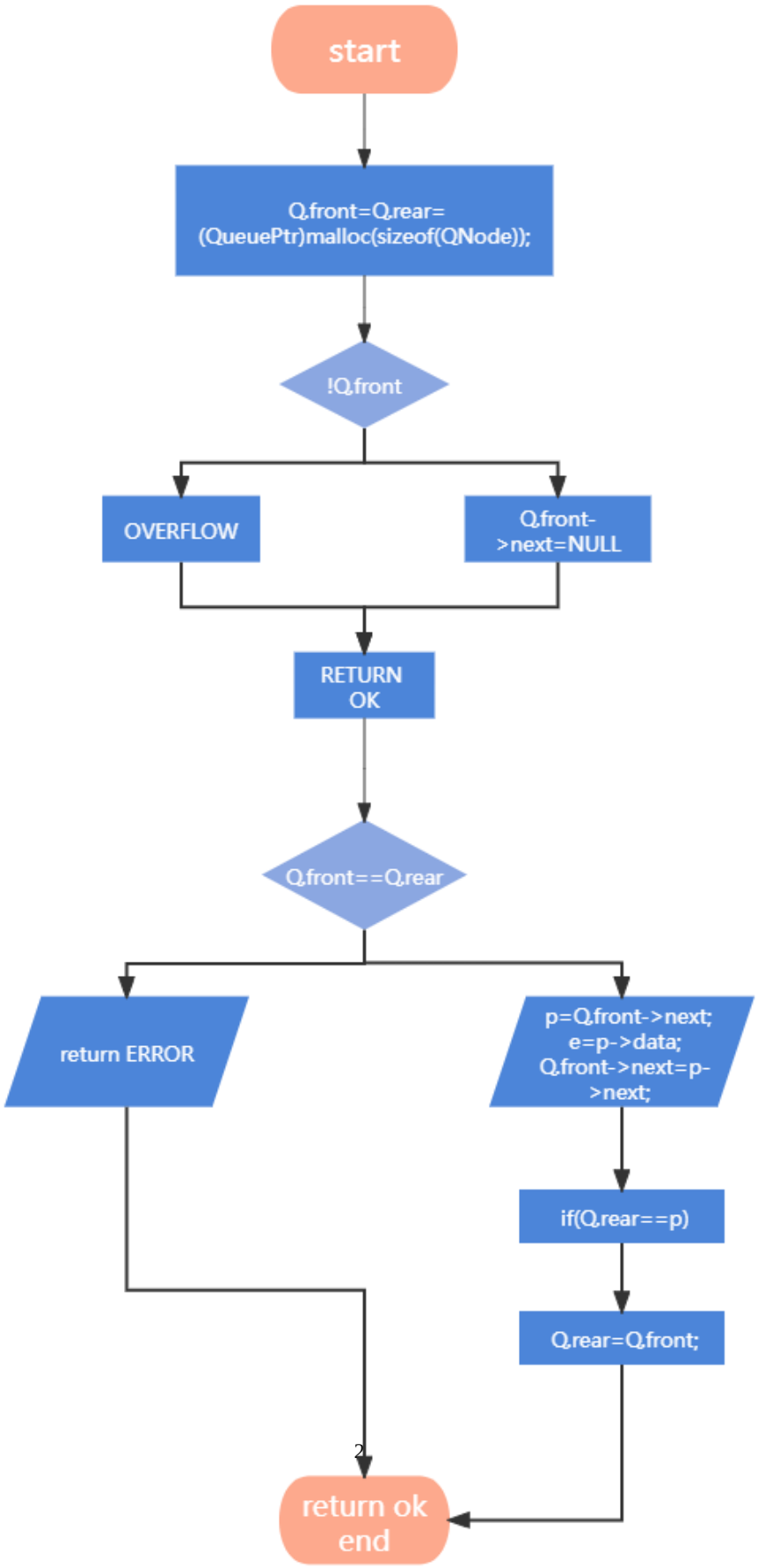
2、(必做题)假设队列中数据元素类型是字符型,
请采用循环队列实现队列的以下基本操作:
(1) Status InitQucuc(&Q)∥构造空队列 Q;
(2) Status EnQucuc(Q,c)元素 c 入队列 Q
(3)Status DeQueue(&,e)∥队列 Q 出队列,元素 e
为 e。

三、数据结构及算法描述

实验一：数据结构：链表 头结点
算法：（1）设有一个队头指针 front 和一个队尾指针 rear
（2）通过队头队尾指针的移动来完成出队和入队操作
（3）初始化时令 front=rear 得到空队列。
（4）入队时申请一个结点后将其链接到队尾结点之后，同时修改队尾指针。
（5）出队时先将队头结点数据域的值送出，然后再修改队头指针，并释放空闲出的内存空间。在主函数中调用并检验。

实验二：数据结构：队列 动态分配内存
算法：（1）构造结构体 QNode，初始化动态分配内存空间
（2）初始化队列，入队列
（3）队列为空时 1%100==1，队列满时（99+1）%100==0，最多容纳 99 个元素
（4）令 rear 在 0-100 内循环
（5）出队列、遍历打印、在主函数中调用并检验

四、详细设计（实验一和实验二的思想流程是一样的，只是用的方法不同）



五、程序代码

实验一：

```
#include <stdio.h>
#include <stdlib.h>
#define OK                  1
#define ERROR               0
#define OVERFLOW            -2

typedef int Status;
typedef char Elemtype;

typedef struct QNode
{
    Elemtype data;
    struct QNode*next;
}QNode,*QueuePtr;

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

Status InitQueue(LinkQueue *Q)//初始化
{
    Q->front=Q->rear=(QueuePtr)malloc(sizeof(QNode));
    if(!Q->front)
        exit(OVERFLOW);
    Q->front->next=NULL;
    return OK;
}

Status DeQueue(LinkQueue *Q,Elemtype *e)//Q 为非空队列，删除 Q 的队头元素，并用 e 返回其值
{
    QueuePtr p;
    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;
    free(p);
    return OK;
}


Status EnQueue(LinkQueue *Q,Elemtype *e)//队列 Q 存在，插入元素 e 为 Q 的队尾元素
{
    QueuePtr p;
    p=(QueuePtr)malloc(sizeof(QNode));
    if(!p)
        exit(OVERFLOW);
    p->data=e;
    p->next=NULL;
    Q->rear->next=p;
    Q->rear=p;
    return OK;
}


int main()
{
    int i,n;
    Elemtype e;
    LinkQueue Q;
    if(InitQueue (&Q))
        printf("队列初始化已经成功！\n");
    else
        printf("队列初始化未成功！\n");
    printf("\n 请输入要入队的元素个数（大于 0 的数字）：\n");
    scanf("%d",&n);
    if(n<=0)
    {
        printf("此输入错误！程序结束！");
        return 0;
    }
    else
    {
        printf("请输入入队元素：\n");
        getchar();
        for(i=0;i<n;i++)
        {
            scanf("%c",&e);
            if(EnQueue(&Q,&e))
```

```
                    printf("第%d 个元素已经成功入队\n",i+1);
                }
    }
    printf("\n 请输入要出队个数（小于等于已入队个数）：\n");
    scanf("%d",&n);
    if(n<=0)
    {
        printf("输入错误导致程序结束！");
        return 0;
    }
    else
    {
        for(i=0;i<n;i++)
        {
            DeQueue(&Q,&e);
            printf("第%d 个出队元素为\t%c\n",i+1,e);
        }
        return 0;
    }
}
```

实验二：

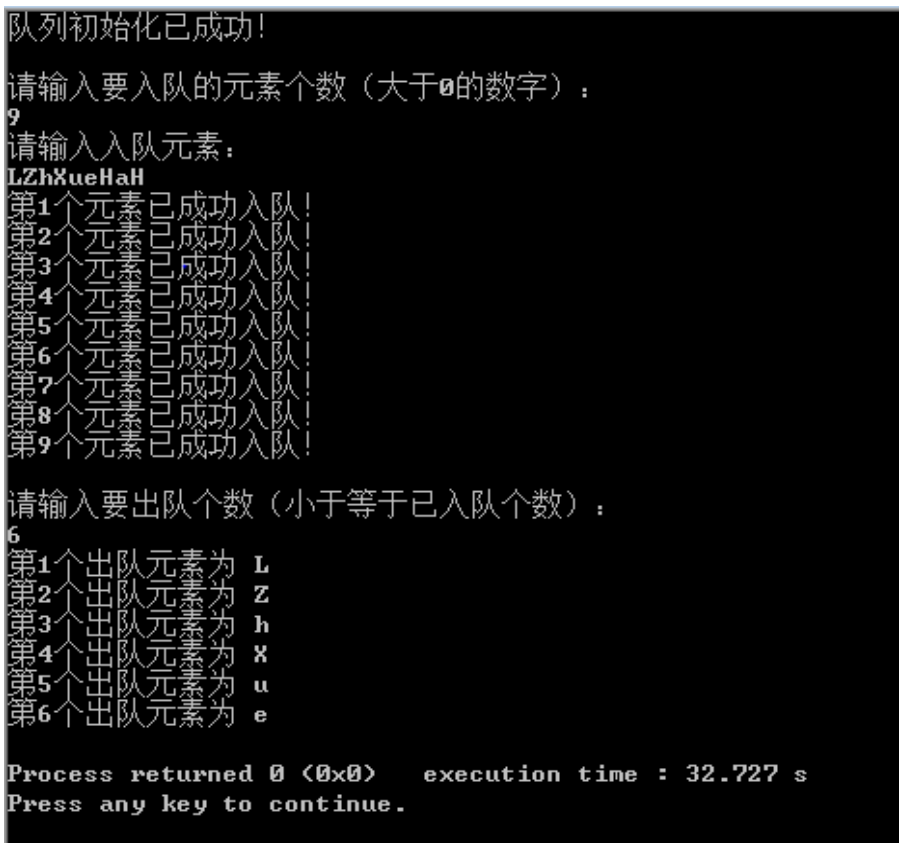
```
#include <stdio.h>
#include <stdlib.h>
#define OK                  1
#define ERROR               0
#define OVERFLOW           -2
#define MAXQSIZE          100

typedef int Status;
typedef char Elemtype;

typedef struct QNode
{
    Elemtype *base;//初始化动态分配内存
    int front;
    int rear;
    struct QNode*next;
}SqQueue;


Status InitQueue(SqQueue*Q)
{
```

```
    Q->base=(Elemtype*)malloc(MAXQSIZE*sizeof(Elemtype));
    if(!Q->base)
        exit(OVERFLOW);
   Q->front=Q->rear=0;
    return OK;
}

Status EnQueue(SqQueue*Q,Elemtype elemt)//入队列
{

    if((Q->rear+1)%MAXQSIZE==(Q->front))
        return ERROR;
    Q->base[Q->rear]=elemt;
    Q->rear=(Q->rear+1)%MAXQSIZE;
    return OK;
}

char DeQueue(SqQueue *Q,Elemtype*e)
{
    if(Q->front==Q->rear)
        printf("输入错误导致程序结束!");
    *e=Q->base[Q->front];
    Q->front=(Q->front+1)%MAXQSIZE;
    return*e;
}

Status OutputQueue(SqQueue Q)
{
    int i;
    for(i=Q.front;i<Q.rear;++i)
    printf("%c",Q.base[i]);
    return OK;
}

int main()
{
    int i,n;
    Elemtype elemt;
    SqQueue Q;
    if(InitQueue (&Q))
        printf("队列初始化已经成功！\n");

    else
        printf("队列初始化未成功！\n");
```

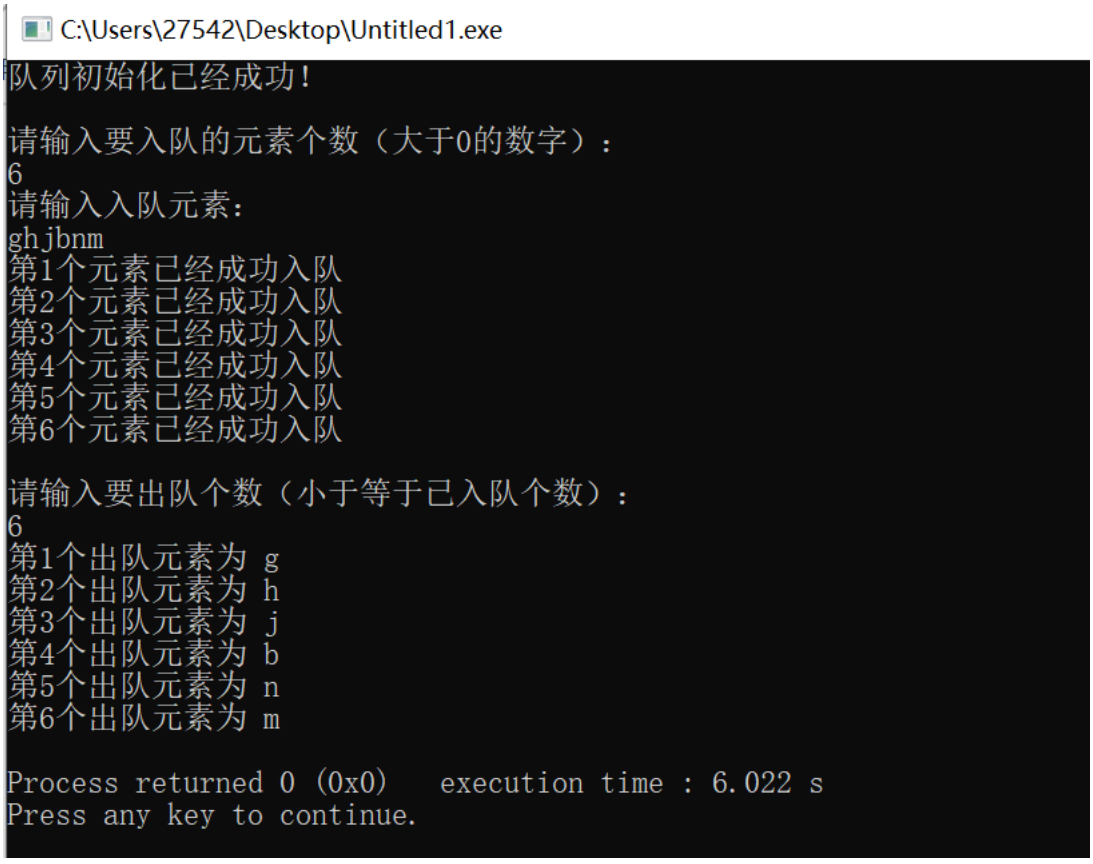
```
    printf("\n 请输入要入队的元素个数（大于 0 的数字）：\n");
    scanf("%d",&n);
    if(n<=0)
    {
        printf("此输入错误！程序结束！");
        return 0;
    }
    else
    {
        printf("请输入入队元素：\n");
        getchar();
        for(i=0;i<n;i++)
        {
            scanf("%c",&elemt);
            if(EnQueue(&Q,elemt))
                printf("第%d 个元素已经成功入队\n",i+1);
              }
    }
    printf("\n 请输入要出队个数（小于等于已入队个数）：\n");
    scanf("%d",&n);
    if(n<=0)
    {
        printf("输入错误导致程序结束！");
        return 0;
    }
    else
    {
        for(i=0;i<n;i++)
        {
            DeQueue(&Q,&elemt);
            printf("第%d 个出队元素为\t%c\n",i+1,elemt);

        }
        return 0;
    }
}
```

六、测试和结果

实验一：



实验二：



七、用户手册

（1）用户按照文字指示进行操作，输入字符串后按回车键结束，进入下一步；

（2）输入时元素类型必须为字符型；

（3）输入出队元素个数不能大于入队元素个数，否则程序会文字提示报错后结束运行，无法再重新输入，只能结束整个程序后重新运行；

（4）实验一的程序可任意输入不限个数的元素，实验二只能输入最多 99 个元素；