附件 2

天 津 商 业 大 学 学 生 实 验 报 告

开课实验室：信息实验室　　开课时间　13 年 9 月 1 日　　实验报告　13 年 10 月 17 日

学院名称	信息工程	年级、专业、班	软件工程 12-01 班	学号	20125041	姓名	王靖伟	同组姓名	
课程名称	数据结构	实验项目名称	实验二　栈和队列			指导教师		黄橡丽	
实验类型	验证 ■	综合 □	设计 □	创新 □				成绩	
教师评语	教师签名： 年　月　日								

实验报告内容一般包括以下几个内容：1、目的要求　2、仪器用具及材料（仪器名称及主要规格、用具名称）　3、实验内容及原理（简单但要抓住要点，写出依据原理）　4、操作方法与实验步骤　5、数据图表格（照片）　6、实验过程原始记录　7 数据处理及结果（按实验要求处理数据、结论）　8、作业题　9、讨论（对实验中存在的问题、进一步的想法等进行讨论）

实验报告内容：

1．实验目的、要求：

（1）掌握栈的特点(先进后出 FILO)及基本操作，如入栈、出栈等，栈的顺序存储结构和链式存储结构，以便在实际问题背景下灵活应用。

（2）掌握队列的特点(先进先出 FIFO)及基本操作,如入队、出队等，队列顺序存储结构、链式存储结构和循环队列的实现，以便在实际问题背景下灵活应用。

2．实验内容（一）：

（1）实验题目

2、循环队列的实现和运算。

（2）算法设计思想（说明整个程序由一个主函数和哪几个函数组成，并给出主要函数的算法设计思想）

1. 构造空队列 Q 函数 Status InitQueue(SqQueue &Q)
2. 判队列是否为空函数 Status QueueEmpty (SqQueue Q)
3. 入队函数 Status EnQueue (SqQueue &Q, QElemType e)
   //插入元素 e 为 Q 的新的队尾元素。
4. 出队函数 Status DeQueue (SqQueue &Q, QElemType &e)
   //若队列不空，则删除 Q 的队头元素，用 e 返回其值，并返回 OK；否则返回 ERROR。
5. 输出循环队列函数 void OutQueue(SqQueue Q)
6. 主函数 void main()

（3）程序清单

```
1    #include <stdio.h>
2    #include <stdlib.h>
3    #define OK 1
4    #define ERROR 0
5    #define OVERFLOW -2
6    typedef int QElemType;
```

```
typedef int Status;
//----- 队列的顺序存储表示 -----
#define    MAXQSIZE    100 //  存储空间的初始分配量
typedef struct {
QElemType    *base;
int    front;
int    rear;
} SqQueue;
//  构造一个空队列 Q
Status InitQueue(SqQueue &Q){
Q.base=(QElemType*)malloc(MAXQSIZE*sizeof(QElemType));
if(!Q.base)exit(OVERFLOW);
Q.front=Q.rear=0;
return OK;
}
//判队列是否为空
Status QueueEmpty (SqQueue Q) {
if(Q.rear==Q.front) return OK;
else return ERROR ;
}
//入队函数
Status EnQueue (SqQueue &Q, QElemType e) {
//请将该算法补充完整
//插入元素 e 为 Q 的新的队尾元素
if((Q.rear+1)%MAXQSIZE==Q.front) return ERROR;//满队列
Q.base[Q.rear]=e;
Q.rear=(Q.rear+1)%MAXQSIZE;
return OK;
}
//出队函数
Status DeQueue (SqQueue &Q, QElemType &e) {
//请将该算法补充完整
//若队列不空，则删除 Q 的队头元素，用 e 返回其值，并返回 OK；
//否则返回 ERROR.
if(Q.front==Q.rear) return ERROR;
e=Q.base[Q.front];
Q.front=(Q.front+1)%MAXQSIZE;
return OK;
}
//输出循环队列函数
void OutQueue(SqQueue    Q)
{       int    n, i;
if(Q.front == Q.rear){
printf("这是一个空队列！");
}
else{
n= (Q.rear- Q.front+ MAXQSIZE) % MAXQSIZE;
i= 1;
```

```
55        while ( i<= n){
56        printf("%6d", Q.base[(Q.front+i-1)% MAXQSIZE] );
57        i++;}
58        printf("\n");
59        }
60        }
61        //主函数
62        void main()
63        {     SqQueue    q;
64        int cord; QElemType a;
65        printf("第一次使用必须初始化！ \n");
66        do{
67        printf("\n                                                主菜单                  \n");
68        printf("    1  初始化循环队列  ");
69        printf("    2  进队一个元素  ");
70        printf("    3  出队一个元素  ");
71        printf("    4  结束程序运行  ");
72        printf("\n--------------------------------------------------------------------------------\n");
73        printf("请输入您的选择( 1, 2, 3, 4)");
74        scanf("%d",&cord);
75        printf("\n");
76        switch(cord)
77        {       case 1:InitQueue(q); //调用初始化算法;
78        OutQueue(q);
79        break;
80        case 2:
81        printf("请输入要插入的数据元素：a=");
82        scanf("%d",&a);
83        EnQueue (q, a);                  //调用进队算法;
84        printf("%d  进队之后的队列：",a);
85        OutQueue(q);
86        break;
87        case 3:
88        DeQueue (q, a);              //调用出队算法;
89        printf("队头元素  %d  出队之后的队列：",a);
90        OutQueue(q);
91        break;
92        case 4:
93        exit(0);
94        }
95        }while (cord<=4);
96        }
```

（3）测试数据

(a) 初始化队列，进队 1、2、3、4，出队，出队，进队 5，结束。
则最终队列：3　4　5

(b) 初始化队列，进队 1、2，出队，进队 3，出队，进队 4，结束。
则最终队列：3　4

（4）实验结果

图1. 测试数据(a)的实验结果片段a.

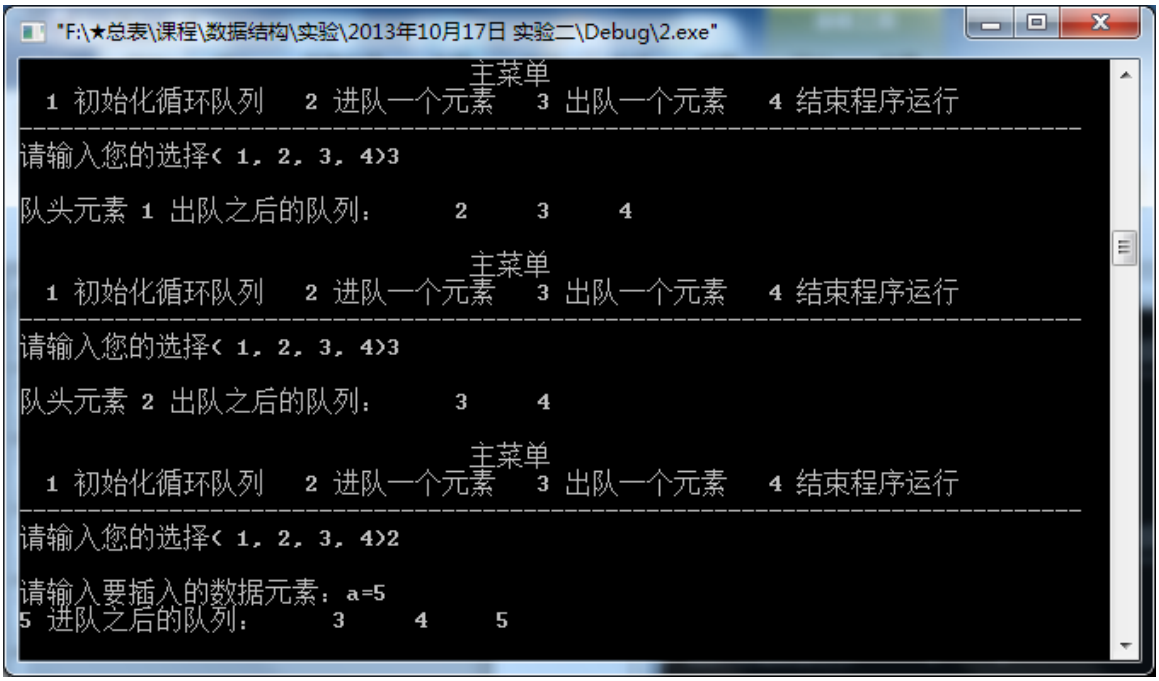

图2. 测试数据(a)的实验结果片段b.

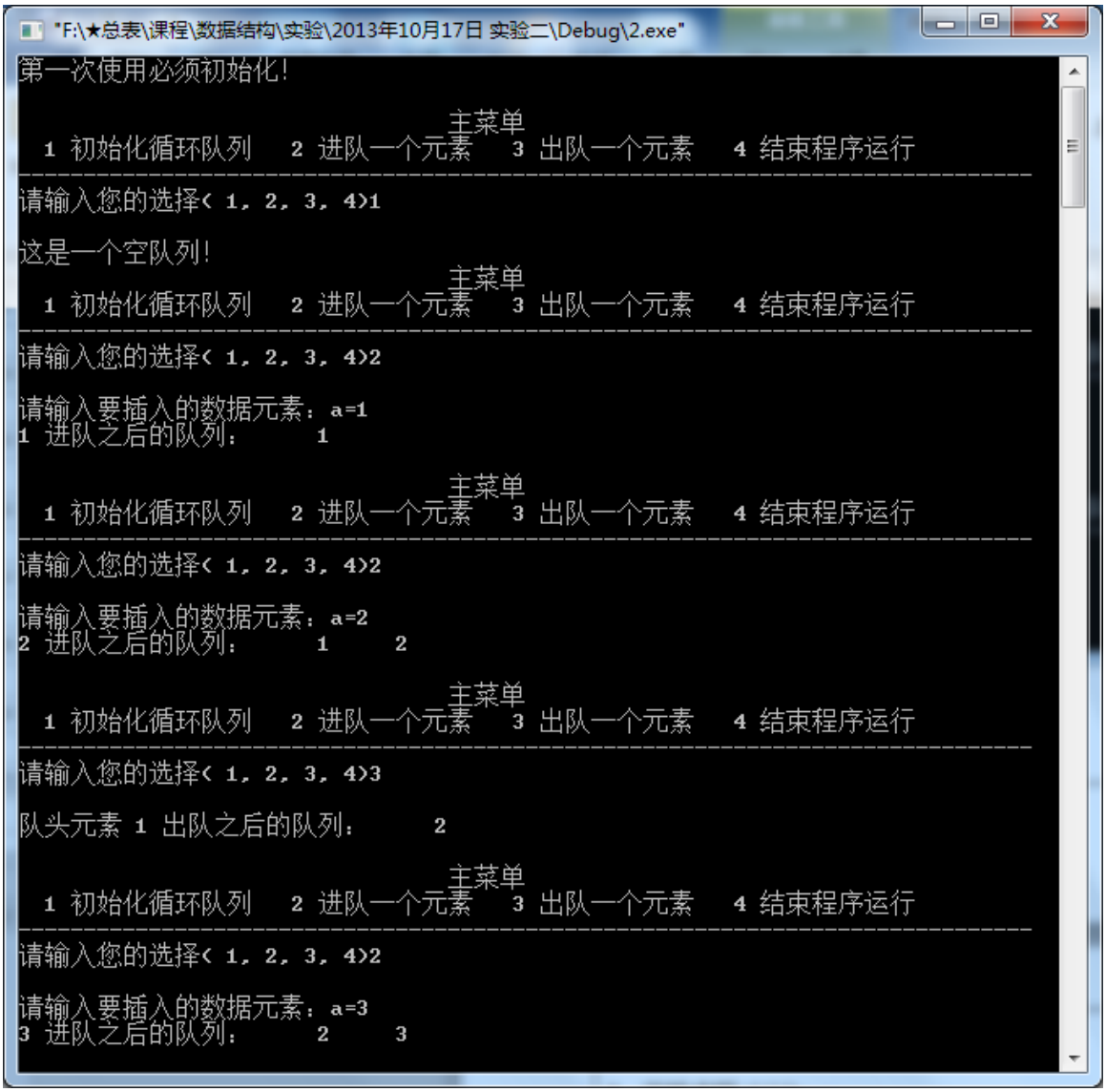

图3. 测试数据(b)的实验结果片段a.

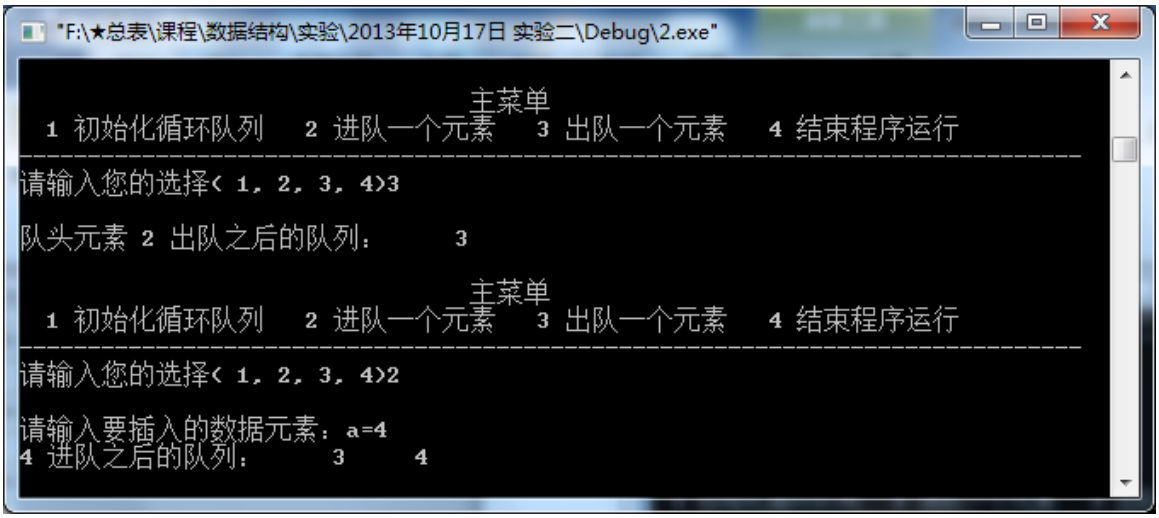

图4. 测试数据(b)的实验结果片段b.

（5）实验收获

入队函数、出队函数的时间复杂度：T(n)=O(1)

入队函数、出队函数的空间复杂度：S(n)=O(1)

3．实验内容（二）：

（1）实验题目

3、栈的运用——十进制转八进制运算。

（2）算法设计思想（说明整个程序由一个主函数和哪几个函数组成，并给出主要函数的算法设计思想）

1. 构造空栈 S 函数 Status InitStack(SqStack &S)
2. 判栈 S 是否为空栈函数 Status StackEmpty(SqStack S)
3. 入栈函数 Status Push (SqStack &S, ElemType e)
   //插入元素 e 为新的栈顶元素；如果栈满，追加储存空间。
4. 出栈函数 Status Pop (SqStack &S, ElemType &e)
   //若栈不为空，则删除 S 的栈顶元素，用 e 返回其值，并返回 OK；否则返回 ERROR。
5. 十进制转八进制函数 void conversion()
6. 主函数 void main()

（3）程序清单

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #define OK 1
4   #define ERROR 0
5   #define OVERFLOW -2
6   typedef int ElemType;
7   typedef int Status;
8   #define  STACK_INIT_SIZE  100
9   #define  STACKINCREMENT  10
10  typedef struct {
11  ElemType  *base;
12  ElemType  *top;
13  int  stacksize;
14  } SqStack;
15  // 构造一个空栈 S
16  Status InitStack(SqStack &S){
17  S.base = (ElemType *)malloc(STACK_INIT_SIZE * sizeof(ElemType));
18  if(!S.base) exit (OVERFLOW);
19  S.top = S.base;
20  S.stacksize = STACK_INIT_SIZE;
21  return OK;
22  }
23  // 判栈 S 是否为空栈
24  Status StackEmpty(SqStack  S){
25  if (S.top==S.base) return OK;
26  else return ERROR;
27  }
28  //入栈函数
29  Status Push (SqStack &S, ElemType e) {
30  //请将该算法补充完整
31  //插入元素 e 为新的栈顶元素
32  if(S.top-S.base>=S.stacksize){//栈满，追加储存空间
```

```
33      S.base=(ElemType*)realloc(S.base,(S.stacksize+STACKINCREMENT)*sizeof(ElemType));
34      if(!S.base) exit(OVERFLOW);//储存分配失败
35      S.top=S.base+S.stacksize;
36      S.stacksize+=STACKINCREMENT;
37      }
38      *S.top++=e;
39      return OK;
40      }
41      //出栈函数
42      Status Pop (SqStack &S, ElemType &e) {
43      //请将该算法补充完整
44      //若栈不为空，则删除 S 的栈顶元素，用 e 返回其值，并返回 OK；
45      //否则反回 ERROR
46      if(S.top==S.base) return ERROR;
47      e=*--S.top;
48      return OK;
49      }
50      //十进制转八进制函数
51      void conversion()
52      {     SqStack S;
53      int n,e;
54      InitStack(S);
55      printf("请输入一个十进制数：");
56      scanf("%d",&n);
57      while(n)
58      {
59      Push(S,n%8);
60      n=n/8;
61      }
62      printf("转换之后的八进制数为：");
63      while(!StackEmpty(S))
64      {
65      Pop(S, e);
66      printf("%d",e);
67      }
68      printf("\n");
69      }
70      void main()
71      {
72      conversion();
73      }
```

（3）测试数据

(a)0D=000　　(b)7D=007　　(c)8D=010　　(d)40D=050

(e)173D=0255　　(f)207D=0317　　(g)16.3D≈020.23　　(h)-55D= -067

（4）实验结果

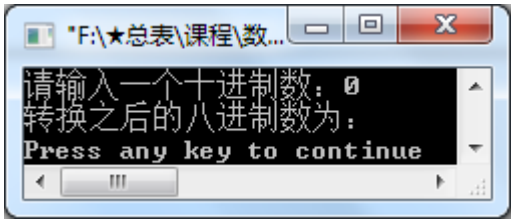

图 5. 测试数据(a)的实验结果

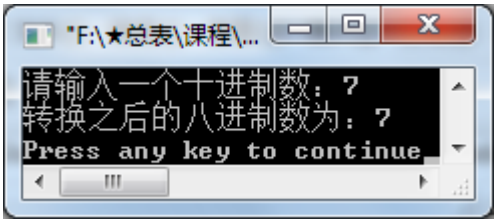

图 6. 测试数据(b)的实验结果

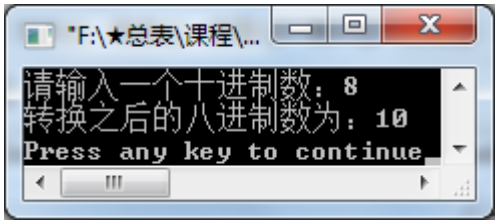

图 7. 测试数据(c)的实验结果

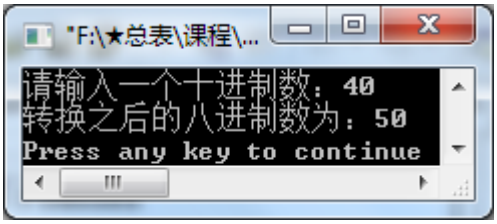

图 8. 测试数据(d)的实验结果

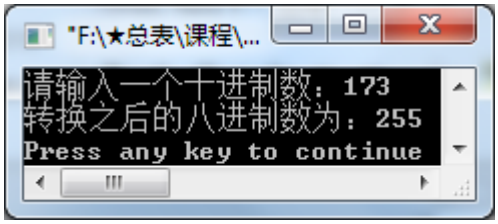

图 9. 测试数据(e)的实验结果

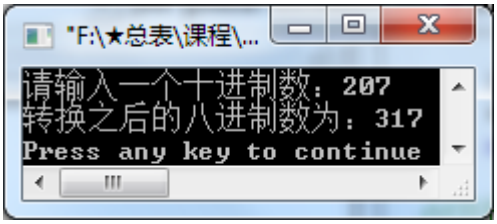

图 10. 测试数据(f)的实验结果

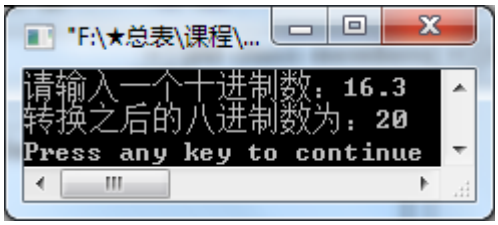

图 11. 测试数据(g)的实验结果

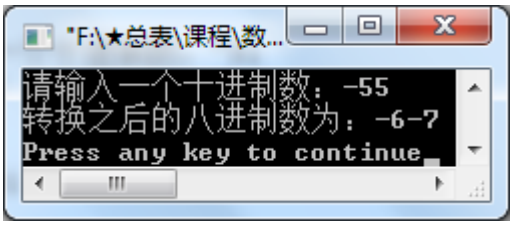

图 12. 测试数据(h)的实验结果

（5）实验收获

入栈函数、出栈函数的时间复杂度：T(n)=O(1)，空间复杂度：S(n)=O(1).

★发现问题：

1.测试数据(a)的结果时发现，结果显示不出来，没有显示理论值。

2.测试数据(g)的结果时发现，结果只能显示出整数部分转换后的结果。

3.测试数据(h)的结果时发现，结果显示在十位和个位上多个一个负号，但数字结果无误。

★原因分析：

本程序只能转换非零正整数。

★问题解决：

通过对本程序的分析，十进制转八进制函数 void conversion()中存在不足，失去对零和负整数以及小数的兼容。遂对 void conversion()函数进行修改。

★修改程序段 void conversion()如下：//带下划线的为添加语句。

```
50    //十进制转八进制函数
51    void conversion()
52    {     SqStack S;
53    int n,e;
54    InitStack(S);// 构造一个空栈 S
55    printf("请输入一个十进制数：");
56    scanf("%d",&n);
57    if(n!=0){
58    if(n>0){
59    while(n){
60    Push(S,n%8);
61    n=n/8;
62    }
63    }else{
64    int temp;
65    temp=0-n;
66    while(temp){
67    Push(S,temp%8);
68    temp=temp/8;
69    }
70    }
71    }else{
72    Push(S,n);
73    }
74    printf("转换之后的八进制数为：");
75    if(n<0){printf("-");}
76    while(!StackEmpty(S))// 判栈 S 是否为空栈
77    {
78    Pop(S, e);
79    printf("%d",e);
80    }
81    printf("\n");
82    }
```

★调试后实验结果：

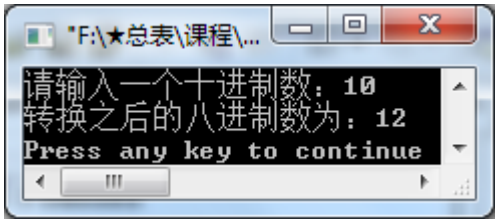

图 13. 测试数据(正整数)的实验结果

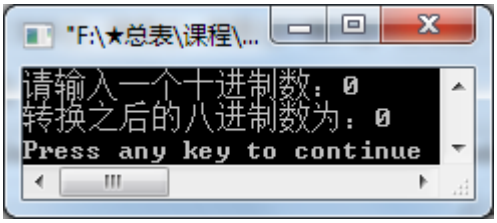

图 14. 测试数据(零)的实验结果

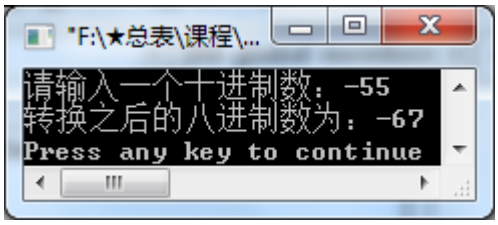

图 15. 测试数据(负整数)的实验结果

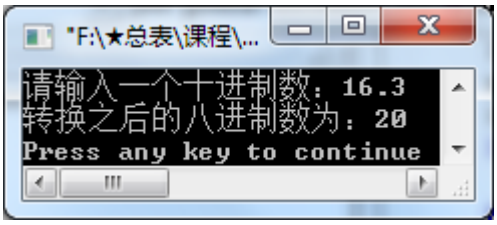

图 16. 测试数据(小数)的实验结果

★结果分析：

修改调试后发现，本程序已经对正整数、负整数、零的完整兼容，但是对于小数的转换还没有实现。由于对小数的实现涉及到对该程序的较大改动，还有待于进一步研究讨论。