# 《数据结构》课程实验报告 2021-2022 学年第 1 学期

_	
姓名	Banban
学号	
班级	
专业	计算机科学与技术
手机	
实验地点	
任课教师	
学院	计算机科学与技术学院

## 实验二 数制转换算法实现

- 1、实验目的
- 1)掌握栈、队列的两类存储结构(顺序存储结构和链式存储结构)的描述方法。
- 2) 掌握在不同结构中实现基本操作的基本方法。
- 3) 掌握在两种结构中实现查找、插入、删除操作的基本方法
- 2、实验内容 编写算法实现任意一个十进制实数转换 r 进制数。
- 3、实验要求

针对实验内容,认真设计算法,上机过程中,能够熟练运用高级语言的程序调试器 DEBUG 调试程序,上机后,认真整理源程序及其注释,完成实验报告(包括源程序、实验结果、算法分析、心得体会等)

### 栈 • 顺序结构

```
#include <stdio.h>
                                         pseqstack start(void)
#include <stdlib.h>
                                         {//创建顺序栈
#define MAXSIZE 100
                                             pseqstack s;
typedef int datatype;
                                         s=(pseqstack)malloc(sizeof(seqstack)
typedef struct {
                                         );
    datatype data[MAXSIZE];
                                             if(s)
    int top;
                                                  s->top=-1;
}seqstack,*pseqstack;
                                             return s;
                                         }
```

```
int empty(pseqstack s)
                                              printf("栈空不能出栈");
{//判断栈是否为空,空栈为1
                                              return 0;//栈空不能出栈
    if(s->top==-1)
                                          }
        return 1;
                                          else
    else
                                          {
        return 0;
                                              *x=s->data[s->top];
}
                                              s->top--;
int push(pseqstack s,datatype x)
                                              return 1;
{//栈顶插入新元素x
                                          }
    if(s->top==MAXSIZE-1)
                                      }
        return 0;//栈满无法入栈
                                      void convert(int x,int r)
    else
                                      {
    {
                                          pseqstack s;
        s->top++;
        s->data[s->top]=x;
                                      s=(pseqstack)malloc(sizeof(seqstack)
        return 1;
                                      );
    }
                                          datatype y;
}
                                          s->top=-1;
int pop(pseqstack s,datatype *x)
                                          while(x){
{//删除栈顶元素,并保存在*x
                                              push(s,x%r);
    if(empty(s))
                                              x=x/r;
    {
                                          }
```

```
while (!empty(s)) {
                                             while (!empty(s)) {
         pop(s,&y);
                                                 pop(s,&y);
         printf("%d",y);
                                                 printf("%d",y);
    }
                                             }
}
void convert1(float x,int r)
                                        }
{
    pseqstack s;
                                        int main()
 输入你想转换的 数字 和 进制 :8.375 2
 1000.110
 Program ended with exit code: 0
s=(pseqstack)malloc(sizeof(seqstack)
);
                                             //pseqstack s;
    datatype y;
                                             int r;
    s->top=-1;
                                             float x,y;
    while(x){
                                             printf("输入你想转换的 数字 和
                                        进制:");
         push(s,(int)(x*r));
         //printf(" %d",(int)(x*r));
                                             scanf("%f%d",&x,&r);
         x=x*r-(int)(x*r);
                                             y=x-(int)(x);
    }
                                             x=(int)x;
```

banban

主意不要雷同

```
printf("\n");
    convert(x,r);
    printf(".");
                                             return 0;
    convert1(y,r);
                                         }
栈•链式存储
#include <stdio.h>
#include <stdlib.h>
                                         s=(plinkstack)malloc(sizeof(linkstack)
#define MAXSIZE 100
                                         );
typedef int datatype;
                                             if(s)
                                                  s->top=NULL;
typedef struct node {
                                             return s;
    datatype data;
                                         }
                                         int empty(plinkstack s)
    struct node *next;
}stacknode,*pstacknode;
                                         {//判空
                                             return (s->top==NULL);
typedef struct {
                                         }
                                         int push(plinkstack s,datatype x)
    pstacknode top;
}linkstack,*plinkstack;
                                         {//进栈
                                             pstacknode p;
plinkstack start(void)
{//创建链式栈
                                         p=(pstacknode)malloc(sizeof(stackno
    plinkstack s;
                                         de));
```

```
if(!p)
                                              return (1);
    {
                                          }
         printf("内存溢出");
                                          void convert(int x,int r)
         return 0;
                                          {
    }
                                              plinkstack s;
    p->data=x;
                                          s=(plinkstack)malloc(sizeof(linkstack)
    p->next=s->top;
                                          );
    s->top=p;
    return (1);
                                              datatype y;
}
                                              s->top=NULL;
int pop(plinkstack s,datatype *x)
                                              while(x){
{//出栈
                                                   push(s,x%r);
    pstacknode p;
                                                   x=x/r;
    if(empty(s))
                                              }
                                              while (!empty(s)) {
    {
         printf("栈空,不能出栈");
                                                   pop(s,&y);
         return 0;
                                                   printf("%d",y);
    }
                                              }
    *x=s->top->data;
                                          }
                                          void convert1(float x,int r)
    p=s->top;
    s->top=s->top->next;
                                          {
    free(p);
                                              plinkstack s;
```

```
push(s,(int)(x*r));
s=(plinkstack)malloc(sizeof(linkstack)
                                                    //printf(" %d",(int)(x*r));
);
                                                    x=x*r-(int)(x*r);
    datatype y;
                                               }
                                               while (!empty(s)) {
    s->top=NULL;
    while(x){
                                               scanf("%f%d",&x,&r);
         pop(s,&y);
         printf("%d",y);
                                               y=x-(int)(x);
    }
                                               x=(int)x;
                                               convert(x,r);
                                               printf(".");
}
int main()
                                               convert1(y,r);
                                               printf("\n");
{
    int r;
                                               return 0;
    float x,y;
                                          }
    printf("输入你想转换的 数字 和
进制:");
```

输入你想转换的 数字 和 进制 :8.375 2

1000.110

Program ended with exit code: 0

### 队列•顺序结构

```
#include <stdio.h>
#include < stdlib.h >
                                         q=(pseqqueue)malloc(sizeof(seqque
#define maxsize 100
                                         ue));
typedef int datatype;
                                              if(q)
                                              {
                                                  q->front=0;
typedef struct{
                                                  q - rear = 0;
    datatype data[maxsize];
                                              }
    int front, rear;
                                              return q;
}seqqueue,*pseqqueue;
                                         }
                                         int empty_seqqueue(pseqqueue q)
pseqqueue init_seqqueue(void)
                                         {
{
                                              if(q\&\&q->front==q->rear)
                                                  return 1;
    pseqqueue q;
                                              else
```

```
if(empty_seqqueue(q))
         return 0;
                                             {
}
                                                 printf("队空");
int in_seqqueue(pseqqueue
q,datatype x)
                                                  return -1;
{
                                             }
                                             else
if((q->rear+1)%maxsize==q->front)
                                             {
    {
         printf("队满");
                                         q->front=(q->front+1)%maxsize;
         return -1;
                                                  *x=q->data[q->front];
    }
                                                  return 1;
    else
                                             }
    {
                                        }
                                         void convert(int x,int r)
q->rear=(q->rear+1)%maxsize;
                                        {
         q->data[q->rear]=x;
                                             seqqueue q;
                                             datatype y;
         return 1;
                                             q.front=q.rear=-1;
    }
}
                                             while (x) {
                                                 in_seqqueue(&q,x%r);
int out_seqqueue(pseqqueue
q,datatype *x)
                                                 x=x/r;
                                             }
{
```

```
while (!empty_seqqueue(&q)) {
         out_seqqueue(&q, &y);
                                         }
         printf("%d",y);
                                         int main()
    }
                                         {
}
                                              float x,y;
void convert1(float x,int r)
                                              int r=2;
                                              printf("输入你想转换的 数字 和
{
                                         进制:");
    seqqueue q;
    datatype y;
                                              scanf("%f%d",&x,&r);
    q.front=q.rear=-1;
                                              y=x-(int)(x);
    while (x>0.001) {
                                              x=(int)x;
         in_seqqueue(&q,(int)(x*r));
                                              convert(x,r);
         x=x*r-(int)(x*r);
                                              printf(".");
    }
                                              convert1(y,r);
    while (!empty_seqqueue(&q)) {
                                              printf("\n");
         out_seqqueue(&q, &y);
         printf("%d",y);
```

输入你想转换的 数字 和 进制 :8.375 2

0001.011

Program ended with exit code: 0

### 队列•链式结构

```
#include <stdio.h>
                                             {
#include < stdlib.h >
                                                 q->front=NULL;
#define maxsize 100
                                                 q->rear=NULL;
typedef int datatype;
                                             }
                                             return 0;
typedef struct node{
                                        }
                                        int empty_linkqueue(plinkqueue q)
    datatype data;
    struct node *next;
                                        {
}qnode,*pqnode;
                                        if(q&&q->front==NULL&&q->rear=
typedef struct{
                                         =NULL)
    pqnode front,rear;
                                                 return 1;
}linkqueue,*plinkqueue;
                                             else
                                                 return 0;
plinkqueue init_linkqueue(void)
{
                                        int in_linkqueue(plinkqueue
    plinkqueue q;
                                        q,datatype x)
                                        {
q=(plinkqueue)malloc(sizeof(linkque
                                             pqnode p;
ue));
    if(q)
                                        p=(pqnode)malloc(sizeof(qnode));
```

```
if(!q)
                                                  return 0;
    {
                                             }
         printf("内存溢出");
                                             *x=q->front->data;
         return 0;
                                             p=q->front;
                                             q->front=q->front->next;
    }
    p->data=x;
                                             free(p);
    p->next=NULL;
                                             if(!q->front)
    if(empty_linkqueue(q))
                                                  q->rear=NULL;
         q->rear=q->front=p;
                                             return 1;
                                         }
    else{
                                         void convert(int x,int r)
         q->rear->next=p;
         q->rear=p;
                                         {
    }
                                             plinkqueue s;
    return 1;
}
                                         s=(plinkqueue)malloc(sizeof(linkque
int out_linkqueue(plinkqueue
                                         ue));
q,datatype *x)
                                             datatype y;
                                             s->front=NULL;
{
    pqnode p;
                                             s->rear=NULL;
    if(empty_linkqueue(q))
                                             while(x){
    {
                                                  in_linkqueue(s,(int)(x%r));
         printf("队空");
                                                  x=x/r;
```

```
}
                                                  printf("%d",y);
    while (!empty_linkqueue(s)) {
                                              }
         out_linkqueue(s,&y);
         printf("%d",y);
                                         }
    }
                                         int main()
}
                                         {
void convert1(float x,int r)
                                              int r;
{
                                              float x,y;
                                              printf("输入你想转换的 数字 和
    plinkqueue s;
                                         进制:");
s=(plinkqueue)malloc(sizeof(linkque
                                              scanf("%f%d",&x,&r);
ue));
                                              y=x-(int)(x);
    datatype y;
                                              x=(int)x;
    s->front=NULL;
                                              convert(x,r);
    s->rear=NULL;
                                              printf(".");
    while(x){
                                              convert1(y,r);
         in_linkqueue(s,(int)(x*r));
                                              printf("\n");
         x=x*r-(int)(x*r);
                                              return 0;
    }
                                         }
    while (!empty_linkqueue(s)) {
         out_linkqueue(s,&y);
                                          输入你想转换的 数字 和 进制 :8.375 2
```

Program ended with exit code: 0

#### 4、实验心得

这次课程实践的心得使我收获颇多: 巩固加深了我对栈、队列的理解,通过 对进制转化的实验提高了我对课程难点有了更多体会,我要在接下来的学习中, 培养独立学习、独立思考的能力,多学多问,以求更快解决所需问题。

下面是我对栈和队列的理解:

栈是一种先进后出的线性结构。只允许在栈的一端进行插入和删除操作,称 为栈顶,栈的另一端称为栈底。栈顶的当前位置是动态变化的,由栈顶指针的位置指示,栈底指向栈的末尾。

顺序栈使用顺序表实现,亦或者说是采用数组实现。 栈还可以采用链式存储结构来存储,称为链式栈。链式栈使用单链表来实现,与顺序栈相比,链栈的结点空间是动态申请的因此一般不存在栈满上溢现象。

队列是一种先进先出的线性结构,只允许在表的一端进行插入和删除操作, 当然:双端队列除外,允许插入的一端称为队尾,允许删除的一端称为队头。由 于在入队和出队的过程中队头指针和队尾指针只增加不减小,致使被删除元素的 空间无法被重新利用,因此,可能会存在这样一种情况:尽管,队列中实际元素 个数远远小于数组大小(队列长度)但可能尾指针已超出数组空间的上界,而不 能进行入队操作,这种现象,称之为"假溢出"。

为了充分利用存储空间,消除这种"假溢出",可以采用的方法是:将为队列分配的空间看成为一个首尾相接的圆环,并称这种队列为循环队列。在循环队列中当队尾指针 rear 达到最大值 Maxsize - 1 时,其队尾指针加 1 操作,使其指向队头指针,这一过程可以使用数学中的取余运算来实现。