《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>
#include <stdlib.h>
#define MAXSIZE 100
typedef int datatype;

typedef struct {
    datatype data[MAXSIZE];
    int top;
}seqstack,*pseqstack;
```

```
pseqstack start(void)
{//创建顺序栈
    pseqstack s;

    s=(pseqstack)malloc(sizeof(seqstack)
);
    if(s)
        s->top=-1;
    return s;
}
```

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

```
    while (!empty(s)) {
        pop(s,&y);
        printf("%d",y);
    }
}
void convert1(float x,int r)
{
    pseqstack s;
```

```
    while (!empty(s)) {
        pop(s,&y);
        printf("%d",y);
    }
}

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)
);
    datatype y;
    s->top=-1;
    while(x){
        push(s,(int)(x*r));
        //printf(" %d",(int)(x*r));
        x=x*r-(int)(x*r);
    }
```

```
{
    //pseqstack s;
    int r;
    float x,y;
    printf("输入你想转换的 数字 和
进制 :");
    scanf("%f%d",&x,&r);
    y=x-(int)(x);
    x=(int)x;
```

```
    convert(x,r);
    printf(".");
    convert1(y,r);
    printf("\n");
    return 0;
}
```

栈·链式存储

```
#include <stdio.h>
#include <stdlib.h>
#define MAXSIZE 100
typedef int datatype;

typedef struct node {
    datatype data;
    struct node *next;
}stacknode,*pstacknode;

typedef struct {
    pstacknode top;
}linkstack,*plinkstack;

plinkstack start(void)
{//创建链式栈
    plinkstack s;
    s=(plinkstack)malloc(sizeof(linkstack)
);
    if(s)
        s->top=NULL;
    return s;
}
int empty(plinkstack s)
{//判空
    return (s->top==NULL);
}
int push(plinkstack s,datatype x)
{//进栈
    pstacknode p;

    p=(pstacknode)malloc(sizeof(stackno
de));
```

```
    if(!p)
    {
        printf("内存溢出");
        return 0;
    }
    p->data=x;
    p->next=s->top;
    s->top=p;
    return (1);
}
int pop(plinkstack s,datatype *x)
{//出栈
    pstacknode p;
    if(empty(s))
    {
        printf("栈空，不能出栈");
        return   0;
    }
    *x=s->top->data;
    p=s->top;
    s->top=s->top->next;
    free(p);
    return (1);
}
void convert(int x,int r)
{
    plinkstack s;

s=(plinkstack)malloc(sizeof(linkstack)
);
    datatype y;
    s->top=NULL;
    while(x){
        push(s,x%r);
        x=x/r;
    }
    while (!empty(s)) {
        pop(s,&y);
        printf("%d",y);
    }
}
void convert1(float x,int r)
{
    plinkstack s;
```

```
s=(plinkstack)malloc(sizeof(linkstack)
);
    datatype y;
    s->top=NULL;
    while(x){
        pop(s,&y);
        printf("%d",y);
    }

}
int main()
{
    int r;
    float x,y;
    printf("输入你想转换的 数字 和
进制 :");
        push(s,(int)(x*r));
        //printf(" %d",(int)(x*r));
        x=x*r-(int)(x*r);
    }
    while (!empty(s)) {

    scanf("%f%d",&x,&r);
    y=x-(int)(x);
    x=(int)x;
    convert(x,r);
    printf(".");
    convert1(y,r);
    printf("\n");
    return 0;
}
```

```
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>
#define maxsize 100
typedef int datatype;


typedef struct{
    datatype data[maxsize];
    int front,rear;
}seqqueue,*pseqqueue;

pseqqueue init_seqqueue(void)
{
    pseqqueue q;
    q=(pseqqueue)malloc(sizeof(seqque
ue));
        if(q)
        {
            q->front=0;
            q->rear=0;
        }
        return q;
}
int empty_seqqueue(pseqqueue q)
{
        if(q&&q->front==q->rear)
            return 1;
        else
```

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

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

```
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>
#define maxsize 100
typedef int datatype;

typedef struct node{
    datatype data;
    struct node *next;
}qnode,*pqnode;

typedef struct{
    pqnode front,rear;
}linkqueue,*plinkqueue;

plinkqueue init_linkqueue(void)
{
    plinkqueue q;

q=(plinkqueue)malloc(sizeof(linkque
ue));
    if(q)
    {
        q->front=NULL;
        q->rear=NULL;
    }
    return 0;
}
int empty_linkqueue(plinkqueue q)
{

if(q&&q->front==NULL&&q->rear=
=NULL)
        return 1;
    else
        return 0;
}
int in_linkqueue(plinkqueue
q,datatype x)
{
    pqnode p;

p=(pqnode)malloc(sizeof(qnode));
```

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

s=(plinkqueue)malloc(sizeof(linkque
ue));
    datatype y;
    s->front=NULL;
    s->rear=NULL;
    while(x){
        in_linkqueue(s,(int)(x%r));
        x=x/r;
```

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

```
输入你想转换的 数字 和 进制 :8.375 2
0001.011
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。