# 第二次作业

目录

[第二次作业 1](#_Toc52822583)

[3.6 1](#_Toc52822584)

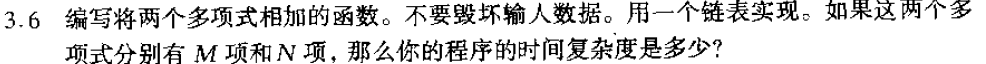
[3.10 3](#_Toc52822585)

[3.12 4](#_Toc52822586)

[3.21 4](#_Toc52822587)

[3.26 4](#_Toc52822588)

## 3.6



用链表不会毁坏输入的数据——数据的内存位置初始化后不会变化，只改变指针

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

typedef int ElemType;

/\*单项链表的声明\*/

typedef struct PolynNode{

int coef; // 系数

int expn; // 指数

struct PolynNode \*next;

}PolynNode,\*PolynList;

/\*正位序(插在表尾)输入n个元素的值，建立带表头结构的单链线性表\*/

/\*指数系数一对一对输入\*/

void CreatePolyn(PolynList &L,int n)

{

int i;

PolynList p,q;

L=(PolynList)malloc(sizeof(PolynNode)); // 生成头结点

L->next=NULL;

q=L;

printf("成对输入%d个数据\n",n);

for(i=1;i<=n;i++)

{

p=(PolynList)malloc(sizeof(PolynNode));

scanf("%d%d",&p->coef,&p->expn); //指数和系数成对输入

q->next=p;

q=q->next;

}

p->next=NULL;

}

//在这个链表上增加1个元素

void AddPolyItem(PolynList &L,int coef, int expn)

{

PolynList p,q;

p=L;

while(p)

{

if(expn==p->expn)

{

p->coef+=coef;

break;

}

p=p->next;

}

if(NULL==p)

{

q=(PolynList)malloc(sizeof(PolynNode));

q->coef=coef;

q->expn=expn;

q->next=NULL;

p->next=q;

}

}

//增加m个元素的链表

void AddPolyM(PolynList &L, int m)

{

for(int i=0; i<m; i++)

{

int coef,expn;

scanf("%d%d",&coef,&expn); //指数和系数成对输入

AddPolyItem(L, coef, expn);

}

}

// 初始条件：单链表L已存在

// 操作结果: 依次对L的每个数据元素调用函数vi()。一旦vi()失败,则操作失败

void PolynTraverse(PolynList L,void(\*vi)(ElemType, ElemType))

{

PolynList p=L->next;

while(p)

{

vi(p->coef, p->expn);

if(p->next)

{

printf(" + "); //“+”号的输出，最后一项后面没有“+”

}

p=p->next;

}

printf("\n");

}

/\*ListTraverse()调用的函数(类型要一致)\*/

void visit(ElemType c, ElemType e)

{

if(c != 0)

{

printf("%dX^%d",c,e); //格式化输出多项式每一项

}

}

void main()

{

PolynList ha;

printf("非递减输入多项式ha， ");

CreatePolyn(ha,2); // 正位序输入n个元素的值

PolynTraverse(ha,visit);

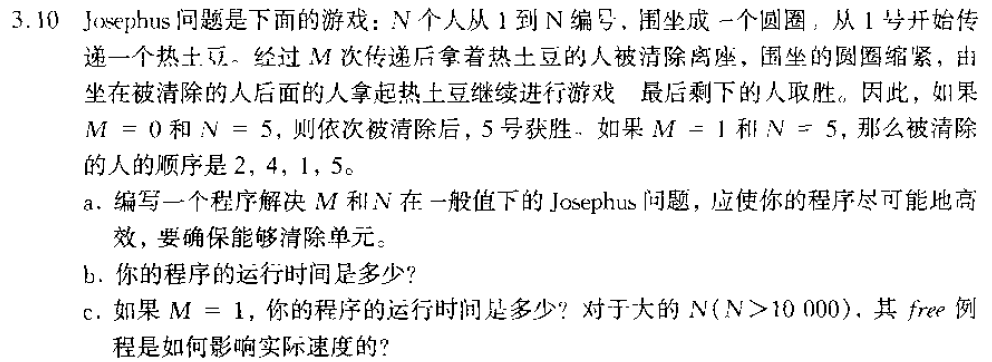
printf("非递减输入多项式ha，\n ");

AddPolyM(ha,2); // 正位序输入n个元素的值

PolynTraverse(ha,visit);

}

## 3.10



#include<stdlib.h>

typedef struct node{

int data;

struct node \*link;

}LNode,\*LinkList;

int main()

{

LinkList p,r,head=NULL;

int i,N,M;

scanf("%d,%d",&N,&M);

//创建N个元素的链表

for(i=1;i<=N;i++)

{

p=(LinkList)malloc(sizeof(LNode));

p->data=i;

if(head==NULL)

{

head=p;

r= head;

}

r->link=p;

r= r->link;

}

p=list;

r=p;

while(p->link!=p)

{ //移动M个位置

for(i=1;i<M+1;i++)

{

r=p;

p=p->link;

}

//第M个出局

r->link=p->link;

printf("%d,",p->data);

free(p);

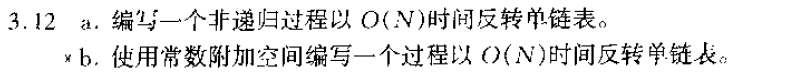
p=r->link;

}

printf("%d",p->data); return(0);

}

## 3.12



using stack

no extra space

/\* Assuming no header and L is not empty. \*/

List ReverseList( List L )

{

Position CurrentPos, NextPos, PreviousPos;

PreviousPos = NULL;

CurrentPos = L;

NextPos = L->Next;

while( NextPos != NULL )

{

CurrentPos->Next = PreviousPos;

PreviousPos = CurrentPos;

CurrentPos = NextPos;

NextPos = NextPos->Next;

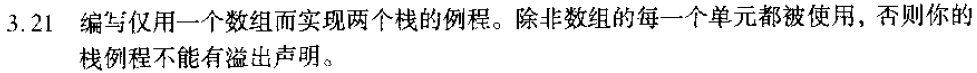
}

CurrentPos->Next = PreviousPos;

return CurrentPos;

}

## 3.21



s[max\_size]=0

top1=0;

top2=max\_size;

bool isEmpty(int index)

{

if(index == 0 && top1 < 1)

return true;

if(index == 1 && top2 > len)

return true;

return false;

}

void push(int index, int data)

{

//已满

if(top1 > top2)

return;

//对栈1操作

if(index == 0)

{

//top

top1++;

s[top1] = data;

}

//对栈2操作

else

{

top2--;

s[top2] = data;

}

}

//出栈，index为0表示对第1个栈进行操作，index为1表示对第2个栈进行操作

int pop(int index)

{

int ret;

//对栈1操作

if(index == 0)

{

//栈1为空

if(top1 < 0)

{

//cout<<"error:underflow"<<endl;

return -1;

}

else

{

ret = s[top1];

top1--;

}

}

//对栈2操作

else

{

//栈2为空

if(top2 > len)

{

//cout<<"error:underflow"<<endl;

return -1;

}

else

{

ret = s[top2];

top2++;

}

}

return ret;

}

## 3.26

