《计算机程序设计》作业 **№-11及第10次上机**

# 作业内容要点： 结构体和链表

【 姓名 马宇骁 学号 PB19151769 】

1. **在计算机上编程程序，加上必要的注释。**
2. **上机实验，经助教检查通过后，复制源码并记录实验结果，完成报告。**
3. **实验报告：记录调试及改错过程；****知识点或方法技巧的收获心得.**

# 1 、用结构体数组实现学生成绩表。

**说明：**

1. **结构体类型定义为：**

**struct student{**

**int stunum; //学号**

**char name[20]; //姓名**

**float examscore; //考试成绩**

**float labscore; //实验成绩**

**float totalmark; //总评成绩**

**};**

1. **在主函数中定义结构体数组，struct student stutable[10];**

**输入如下十个学生的成绩数据，每个学生信息包括 学号 、姓名 、考试成绩 ，实验成绩。同时计算每个学生的总评成绩（ =考试成绩\*60% + 实验成绩\*40%）并保存至每个结构体的totalmark。输入格式如下：**

71250 李霞 95 82

69753 李友友 88 86

12254 东方亮 87 88

61256 张男 73 85

30258 孙杰 25 88

11260 柯以乐 82 76

33262 谢涛 91 85

29263 叶林 80 75

22483 陈翔 80 76

71525 王子 71 88

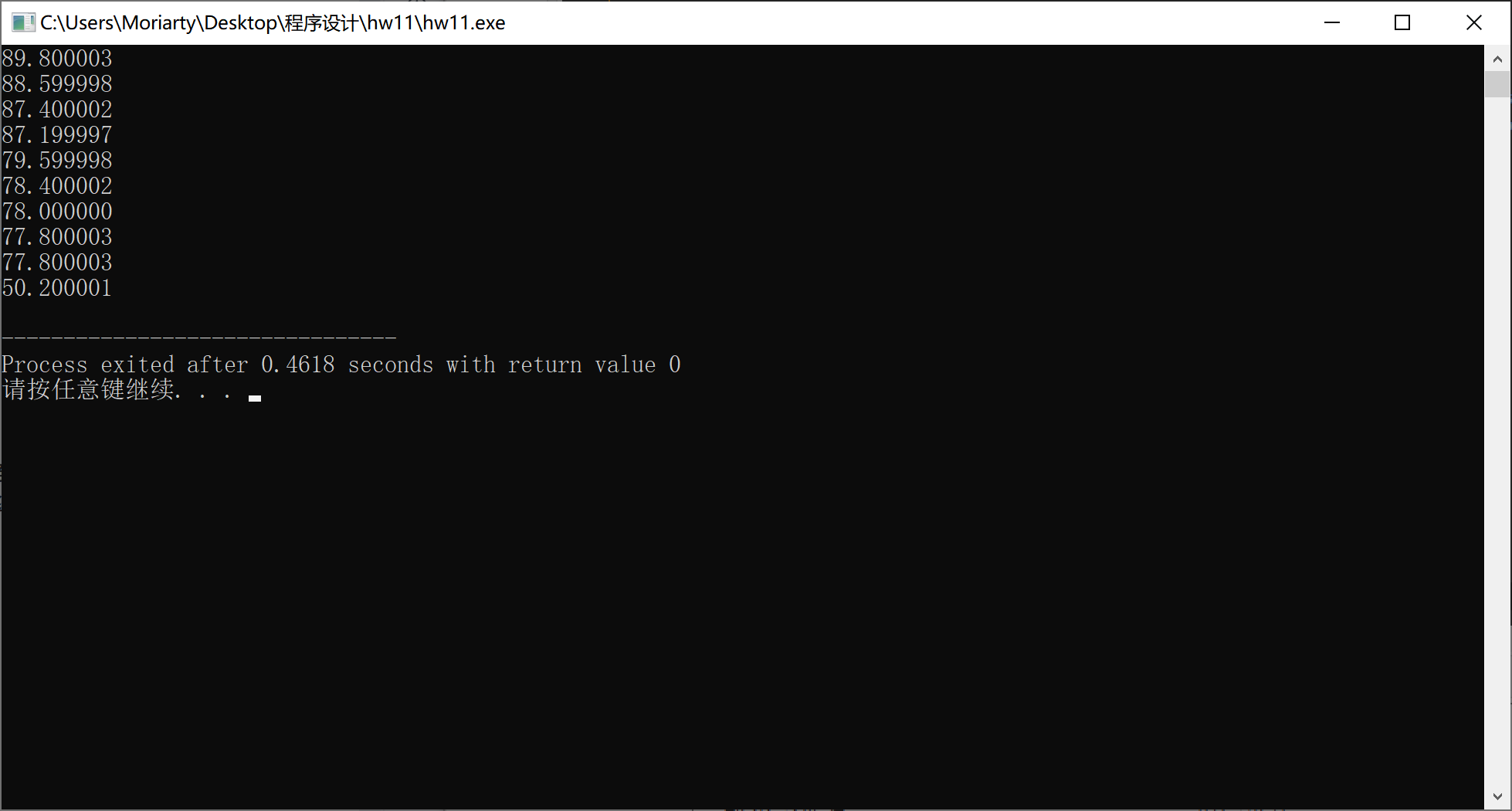
1. **在主函数中定义一个结构体指针数组，struct student \*parrray[10] ; 使其每一个指针指向上述结构体数组中的一个元素； 按总评成绩从高到低的顺序，对指针数组parray进行排序，数组stutable保持不变，可避免结构体数组元素之间的交换移动。按总评成绩从高到低的顺序输出排序之后的全部学生成绩。**

**输入输出样例：略**

1. 【源码】

* #include<stdio.h>
* #include <string.h>
* struct student{
* int stunum; //学号
* char name[20]; //姓名
* float examscore; //考试成绩
* float labscore; //实验成绩
* float totalmark; //总评成绩
* };
* int main(){
* struct student stutable[10];
* struct student \*parrray[10];
* freopen("hw11.1-student.txt","r",stdin);
* for(int i=0;i<10;i++){
* scanf("%d %s %f %f",&stutable[i].stunum,&stutable[i].name,&stutable[i].examscore,&stutable[i].labscore);
* stutable[i].totalmark = 0.6\*stutable[i].examscore + 0.4\*stutable[i].labscore;
* parrray[i] = &stutable[i];
* }
* //bubble
* struct student \*tmp;
* for(int i=0;i<10-1;i++){
* for(int j=0;j<10-i-1;j++){
* if(parrray[j]->totalmark < parrray[j+1]->totalmark){
* tmp = parrray[j];
* parrray[j] = parrray[j+1];
* parrray[j+1] = tmp;
* }
* }
* }
* for(int i=0;i<10;i++){
* printf("%f\n",parrray[i]->totalmark);
* }
* }

1. 【运行结果】

* 

1. 【实验报告】

* 创建结构体，利用文件中的信息读取进结构体；利用结构体指针指向已有结构体，对其冒泡排序，输出，过程中并未出现问题。

**2. 用链表实现学生成绩表管理。**

**接上题。**

1. **结构体类型定义修改为：**

**struct student{**

**int stunum; //学号**

**char name[20]; //姓名**

**float examscore; //考试成绩**

**float labscore; //实验成绩**

**float totalmark; //总评成绩**

**struct student \* next; //下一个结点**

**};**

1. **编写函数实现建立链表：struct student \* create( int n), n是学生人数。函数中输入n个学生的信息，同时计算总评成绩，按照总评成绩从高到低的方式形成有序链表。返回链表头指针。**
2. **编写函数 struct student \* delete(struct student \* head, int stunum), 将学号为stunum的结点删除；返回链表头指针。**
3. **编写函数struct student \* insert( struct student \* head), 插入一个新的结点到链表中，并保持按总评成绩从高到低有序。返回链表头指针。**
4. **在主函数中分别调用上述函数，建立链表的10个学生数据同第一题。删除结点时的测试数据可以是现有的学号、也可以是不存在的学号—函数应输出提示未找到并返回原有头指针。新增结点时数据为任意与现有结点不同的值。在主函数中输出每次函数调用后的链表内容。**

**输入输出样例：略**

**#include<stdio.h>**

**#include <string.h>**

**#include<stdlib.h>**

**typedef struct LNode{**

**int stunum; //学号**

**char name[20]; //姓名**

**float examscore; //考试成绩**

**float labscore; //实验成绩**

**float totalmark; //总评成绩**

**struct LNode \* next; //下一个结点**

**}LNode,\*Link;**

**void print(Link head){**

**Link p;**

**p = head;**

**while(p != NULL){**

**printf("%d %s %f %f %f\n",p->stunum,p->name,p->examscore,p->labscore,p->totalmark);**

**p = p->next;**

**}**

**}**

**Link create(int n){**

**Link p,l,tmp;**

**p = (Link)malloc(sizeof(LNode));**

**p->next = NULL;**

**l = p;**

**freopen("hw11.1-student.txt","r",stdin);**

**for(int i=0;i<n-1;i++){**

**scanf("%d %s %f %f",&p->stunum,&p->name,&p->examscore,&p->labscore);**

**p->totalmark = 0.6\* (p->examscore) + 0.4\* (p->labscore);**

**tmp = (Link)malloc(sizeof(LNode));**

**p->next = tmp;**

**p = p->next;**

**p->next = NULL;**

**}**

**scanf("%d %s %f %f",&p->stunum,&p->name,&p->examscore,&p->labscore);**

**p->totalmark = 0.6\* (p->examscore) + 0.4\* (p->labscore);**

**fclose(stdin);**

**//bubble**

**Link head;**

**head = (Link)malloc(sizeof(LNode));**

**head->next = l;**

**Link tail,q1,q2;**

**tail = NULL;**

**while((head->next->next) != tail){**

**q1 = head;**

**q2 = head->next;**

**while(q2->next != tail){**

**if((q2->totalmark) < (q2->next->totalmark)){**

**q1->next = q2->next;**

**q2->next = q2->next->next;**

**q1->next->next = q2;**

**q2 = q1->next;**

**}**

**q2 = q2->next;**

**q1 = q1->next;**

**}**

**tail = q2;**

**}**

**tmp = head;**

**head = head->next;**

**free(tmp);**

**freopen("CON", "r", stdin);**

**return head;**

**}**

**Link delete1(Link head,int stunum){**

**Link h,tmp,l;**

**h = (Link)malloc(sizeof(LNode));**

**h->next = head;**

**l = h;**

**while(h->next->next != NULL){**

**if(h->next->stunum == stunum){**

**tmp = h->next;**

**h->next = h->next->next;**

**free(tmp);**

**return l->next;**

**}**

**h = h->next;**

**}**

**if(h->next->next == NULL){**

**if(h->next->stunum == stunum){**

**tmp = h->next;**

**h->next = NULL;**

**free(tmp);**

**return l->next;**

**}**

**}**

**else{**

**printf("没找到\n");**

**return l->next;**

**}**

**}**

**Link insert(Link head){**

**Link p,l;**

**l = head;**

**p = (Link)malloc(sizeof(LNode));**

**scanf("%d %s %f %f",&p->stunum,&p->name,&p->examscore,&p->labscore);**

**p->totalmark = 0.6\* (p->examscore) + 0.4\* (p->labscore);**

**if(p->totalmark > head->totalmark){**

**p->next = head;**

**return p;**

**}**

**while(l->next != NULL){**

**if((p->totalmark <= l->totalmark) && (p->totalmark >= l->next->totalmark)){**

**p->next = l->next;**

**l->next = p;**

**return head;**

**}**

**l = l->next;**

**}**

**if((p->totalmark <= l->totalmark)){**

**l->next = p;**

**return head;**

**}**

**}**

**int main(){**

**int m,n=10,s;**

**Link l = (Link)malloc(10\*sizeof(LNode));**

**Link p;**

**while(1){**

**system("CLS");**

**printf("请输入功能：\n1.建立链表\n2.删除结点\n3.插入结点\n0.退出\n");**

**scanf("%d",&m);**

**switch(m){**

**case 0: free(l);return 0;**

**case 1:**

**getchar();**

**l = create(n);**

**p = l;**

**print(p);**

**break;**

**case 2:**

**scanf("%d",&s);**

**l = delete1(l,s);**

**p = l;**

**print(p);**

**break;**

**case 3:**

**l = insert(l);**

**p = l;**

**print(p);**

**break;**

**default:**

**printf("没这个功能\n");**

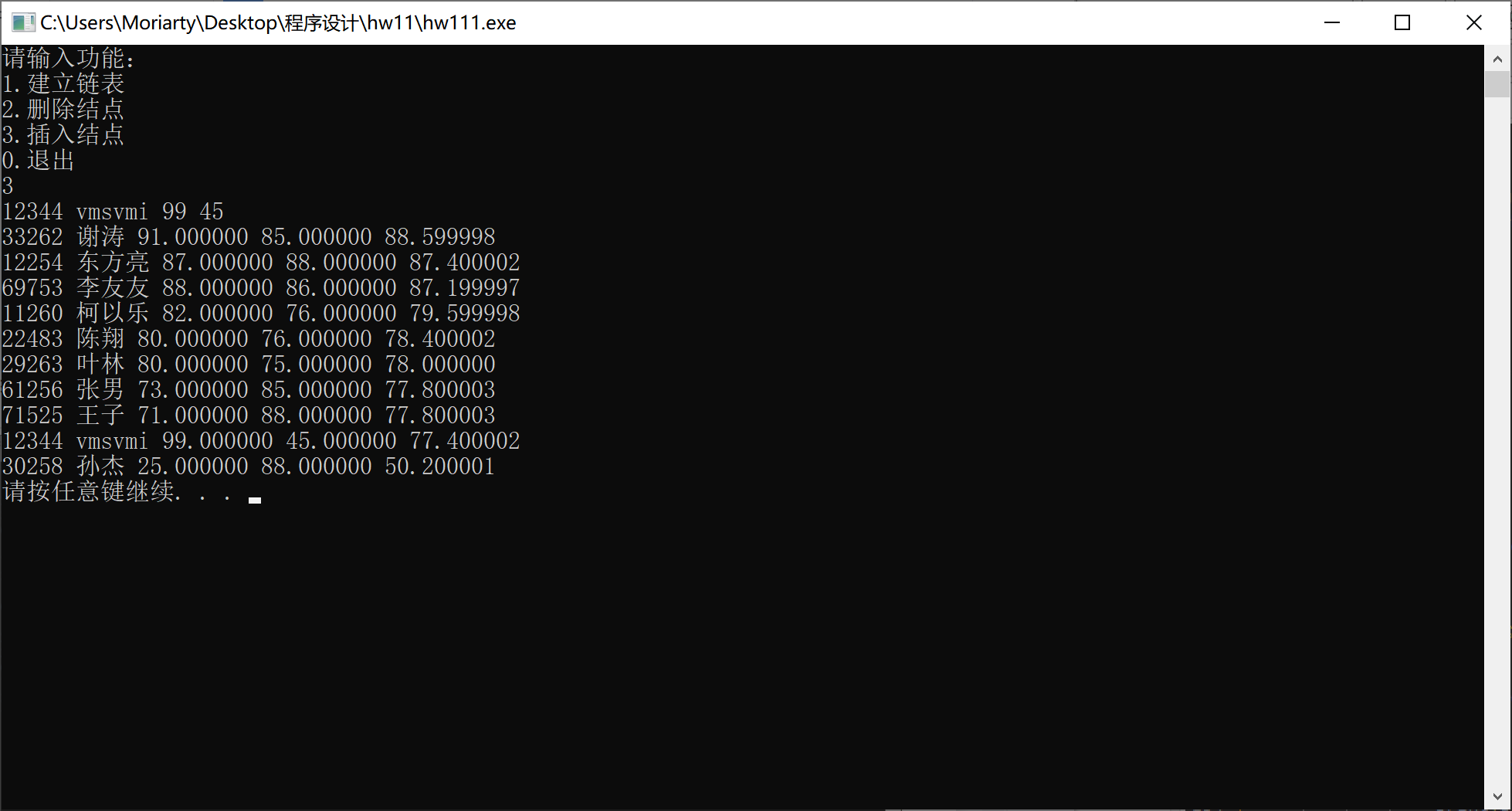
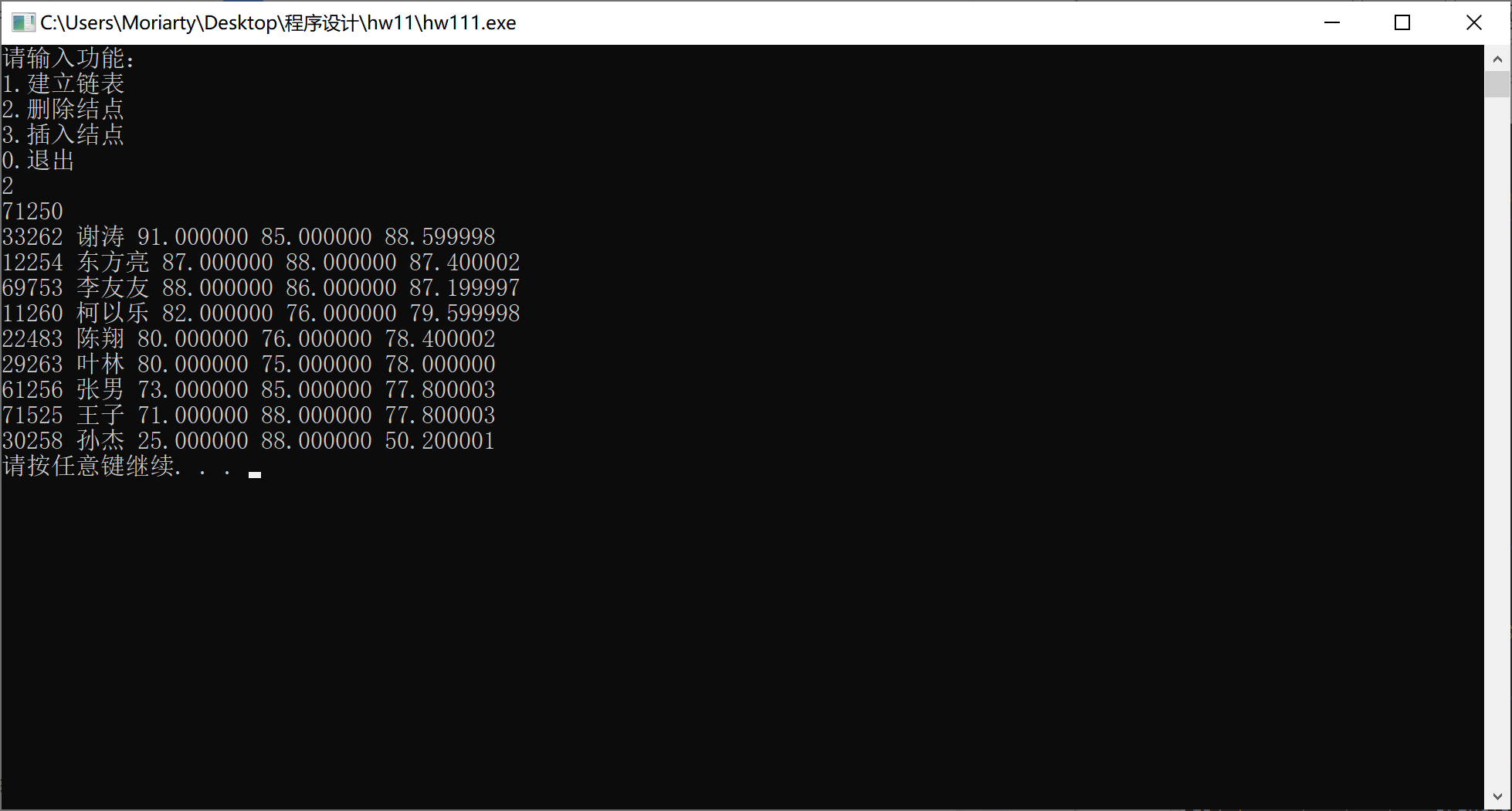
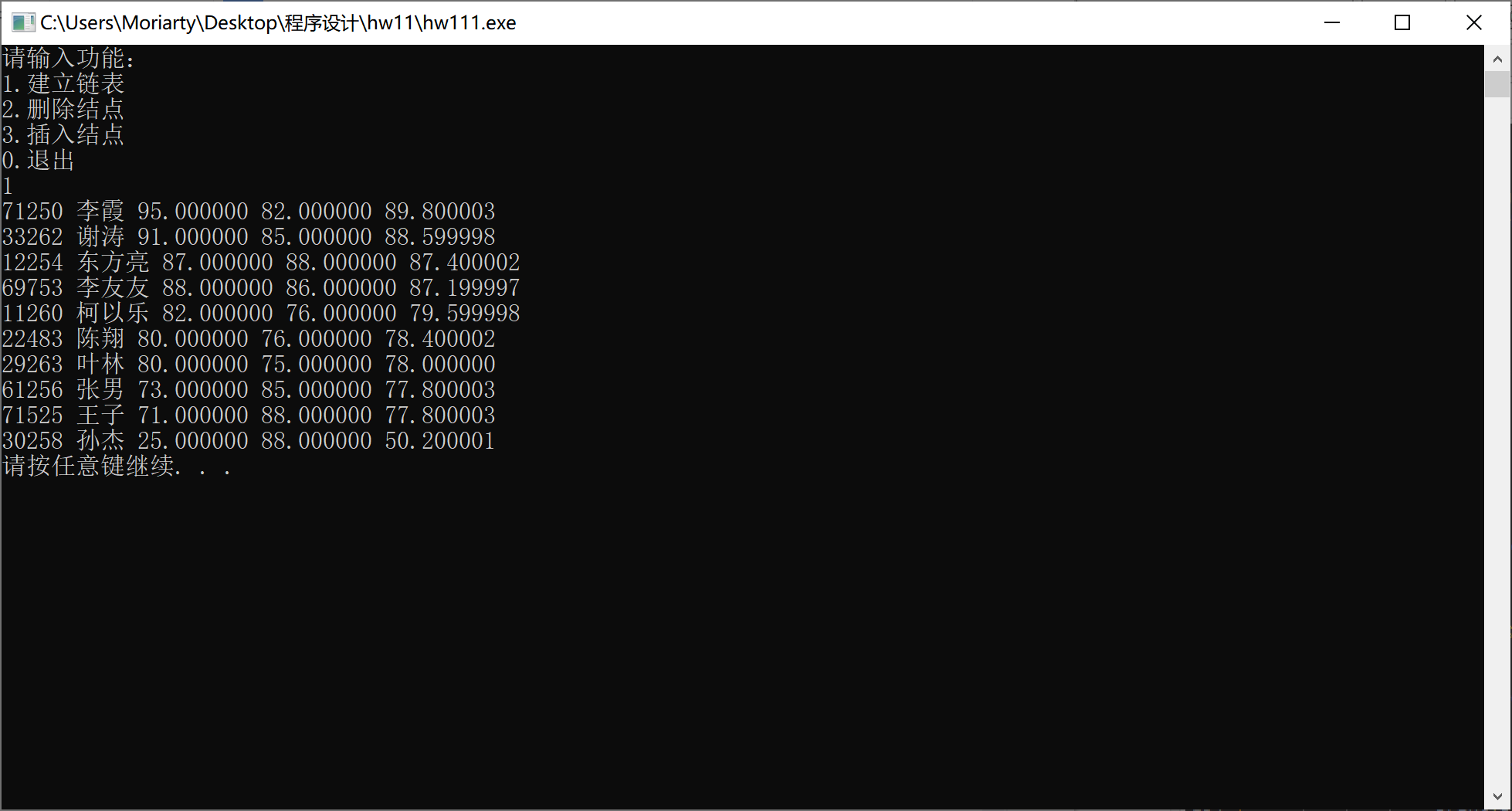
**break;**

**}**

**system("PAUSE");**

**}**

**}**



考察了链表的读取（遇到了很多bug发现直接输入没有问题但freopen的时候回出现频闪不能进行下一步，经过排查，使用freopen("CON", "r", stdin);重定向回屏幕解决），链表的冒泡排序（建立头指针之后判断尾部在对其中的结点进行交换的操作），链表的结点删除（记得free）和加入。

**3 、 用链表实现求两个多项式的和。**

**说明：**

1. 一个多项式可以表示为二元组序列{(a1,e1), (a2,e2), … (an,en)}, 其中ai表示第i项的系数（非零值）， ei表示第i项的指数。
2. 编写函数建立多项式链表实现一个多项式的输入，按指数从高到低有序，返回链表的头指针。

3) 编写函数实现两个多项式相加，返回结果多项式链表的头指针。

4) 编写函数输出一个多项式的二元组序列。

5) 在main函数中分别调用上述函数，实现输入两个多项式，求出它们的和并输出结果。

6) 输入数据分2行，每行分别先给出多项式非零项的个数，再输入每一对非零项系数和指数（假设为绝对值均为不超过10000的整数）。数字间仅以空格分隔。

7) 为简化处理，限定系数与指数都为整数。

**链表结点数据结构可定义为：**

**struct PolyNode{**

**int a; //系数**

**int e; //指数**

**PolyNode \* next; //指向下一个结点**

**};**

**输入样例:**

**4 3 4 -5 2 6 1 -2 0**

[注]表示 3x4-5x2+6x-2

**3 5 20 -7 4 3 1**

[注]表示 5x20-7x4+3x

**输出样例:**

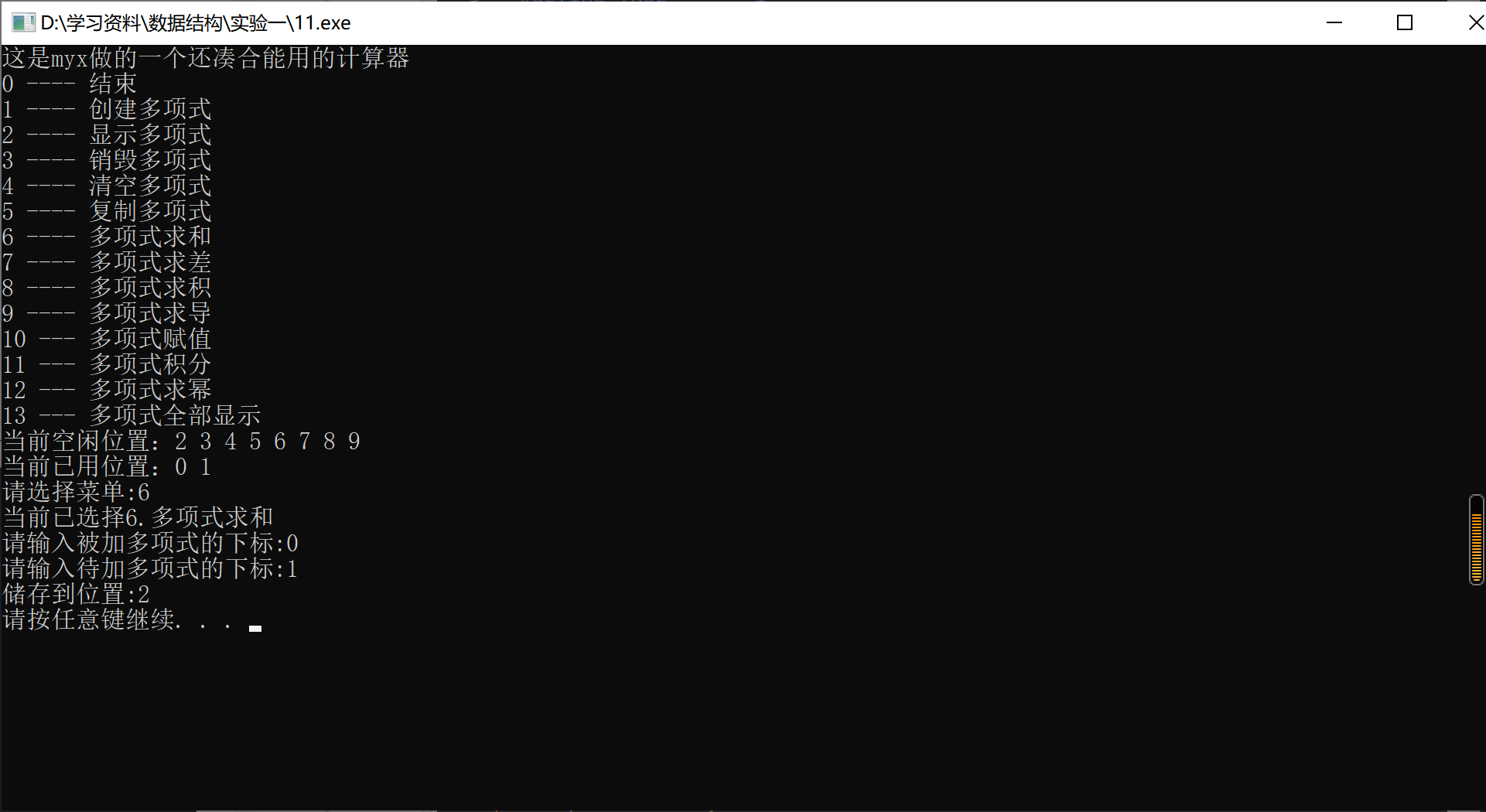
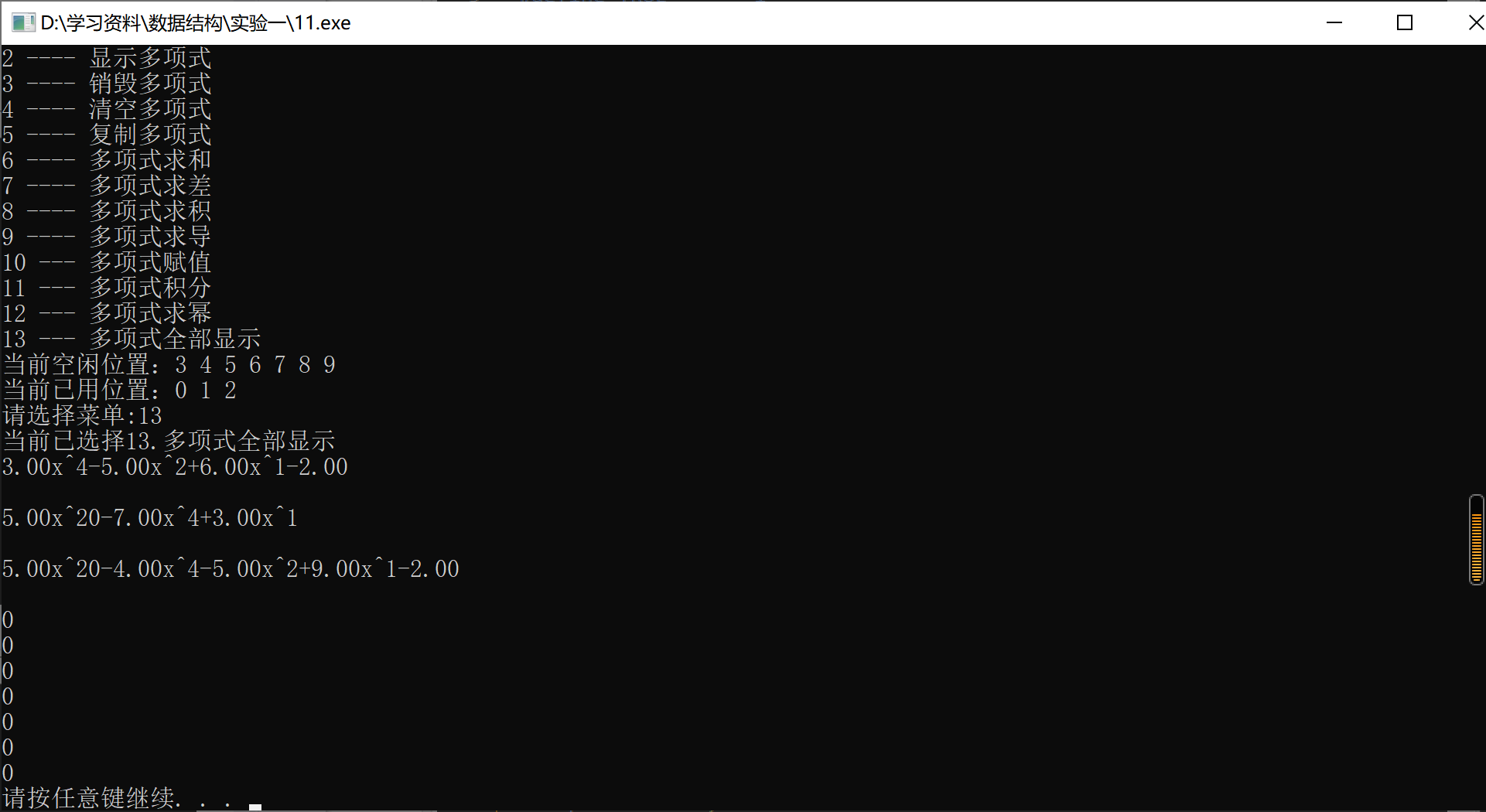
**5 20 -4 4 -5 2 9 1 -2 0**

[注]表示 5x20-4x4-5x2+9x-2

1. 【源码】

* #include<stdio.h>
* #include<string.h>
* #include<stdlib.h>
* #include<math.h>
* #define TRUE 1
* #define FALSE 0
* #define OK 1
* #define ERROR 0
* #define INFEASIBLE -1
* typedef int Status;
* #define Abort(msg) { printf("aborting:%s\n",msg);exit(0);}
* #define N 10
* typedef struct{
* float coef;
* int expn;
* }/\*term,\*/ ElemType;//项。元素类型
* typedef struct LNode{
* ElemType data;
* struct LNode \*next;
* }LNode, \*Link, \*Position;//link==\*lNode
* typedef struct{
* Link head;
* int len;
* }LinkList;//多项式
* typedef LinkList Polynomial;
* typedef Status (\*cmpf)(ElemType, ElemType);
* typedef Status (\*visf)(ElemType);
* Status MakeNode(Link &p, ElemType e){
* p = (Link)malloc(sizeof(LNode));
* if(!p)return ERROR;
* p->data = e;
* p->next = NULL;
* return OK;
* }
* void FreeNode(Link &p){
* free(p);
* }
* Status InitList(LinkList &L){
* L.head = (Link)malloc(sizeof(LNode));
* if(!L.head)exit(OVERFLOW);
* L.head->next = NULL;//为头结点申请空间并判断非空
* return OK;
* }
* Status DestroyList(LinkList &L){
* Link tail;
* while(L.head){
* tail = L.head->next;
* free(L.head);
* L.head = tail;
* }
* return OK;
* }
* Status ClearList(LinkList &L){
* Link p = L.head->next;
* while(L.head->next){
* p = L.head->next;
* L.head->next = p->next;
* free(p);
* }
* return OK;
* }
* Status InsFirst(Link h, Link s){
* if(!h)return ERROR;
* s->next = h->next;
* h->next = s;
* return OK;
* }
* Status DelFirst(Link h, Link &q){
* Link p = h->next;
* if(!p)return ERROR;
* h->next = p->next;
* q = p;
* return OK;
* }
* Status Append(LinkList &L, Link s){
* Link tail = L.head;
* while(tail->next)tail = tail->next;//连接两个链表
* tail->next = s;
* }
* ElemType GetCurElem(Link p){
* return p->data;//获取结点值
* }
* Status SetCurElem(Link &p, ElemType e){
* if(!p)return ERROR;
* p->data = e;// 赋结点值
* return OK;
* }
* Position GetHead(LinkList L){
* return L.head;
* }
* Status ListEmpty(LinkList L){
* return !L.head->next;//判断表是否是空
* }
* Position NextPos(LinkList L, Link p){
* return p->next;
* }
* Status LocateElem(LinkList &L, ElemType e, Position &q, cmpf compare){
* q = L.head;
* Link p = L.head;
* while(p != NULL){
* if(compare(p->data, e) <= 0)q = p; //为后面四则运算的项找位置
* if(!compare(p->data, e)){
* q = p;
* return TRUE;
* }
* p = p->next;
* }
* return FALSE;
* }
* int cmp(ElemType a, ElemType b){
* if(a.expn - b.expn < 0)return 1;
* if(a.expn - b.expn > 0)return -1;
* return 0;
* }
* void PrintElem(ElemType e){
* if(!e.expn)printf("%.2f", e.coef);
* else printf("%.2fx^%d", e.coef, e.expn);
* }
* void PrintPoly(Polynomial P){
* Link p = P.head->next;
* if(!p){
* printf("0\n");
* return;
* }
* while(p){
* if(p != P.head->next && p->data.coef > 0)printf("+");
* PrintElem(p->data);
* p = p->next;
* }
* putchar('\n');
* }
* void CreatPoly(Polynomial &P, int m){
* Link s, q;
* InitList(P);
* Link h = GetHead(P);
* ElemType e = {0.0, -1};
* SetCurElem(h, e);
* for(int i = 1; i <= m; i++){
* scanf("%f%d", &e.coef, &e.expn);
* if(!LocateElem(P, e, q, cmp)){
* if(MakeNode(s, e)) InsFirst(q, s);
* PrintElem(e);
* }
* }
* }
* Status InitPoly(Polynomial &P){
* InitList(P);
* Link h = GetHead(P);
* ElemType e = {0.0, -1};//初始化多项式
* SetCurElem(h, e);
* return OK;
* }
* Status CopyPoly(Polynomial &Pa, Polynomial Pb){
* if(Pa.head)ClearList(Pa);
* else InitPoly(Pa);
* Link pb = NextPos(Pb, GetHead(Pb)), pa, pt = GetHead(Pa);
* while(pb){
* pa = (Link)malloc(sizeof(LNode));
* if(!pa)return ERROR;
* pa->data = pb->data;
* pt->next = pa;
* pt = pa;
* pb = pb->next;
* }
* pt->next = NULL;
* return OK;
* }
* Polynomial AddPoly(Polynomial Pa, Polynomial Pb){
* Polynomial P;
* InitPoly(P);
* Link p = GetHead(P);
* Link qa = NextPos(Pa, GetHead(Pa)), qb = NextPos(Pb, GetHead(Pb));//多项式第一项
* ElemType a, b;
* float sum = 0;
* while(qa && qb){
* a = GetCurElem(qa); b = GetCurElem(qb);
* switch(cmp(a, b)){
* case -1:
* p->next = (Link)malloc(sizeof(LNode));
* p = NextPos(P, p);
* p->data = qa->data;
* qa = NextPos(Pa, qa);
* break;
* case 0:
* sum = a.coef + b.coef;
* if(sum != 0.0){
* p->next = (Link)malloc(sizeof(LNode));
* p = NextPos(P, p);
* p->data.coef = sum;
* p->data.expn = qa->data.expn;
* }
* qb = NextPos(Pb, qb);
* qa = NextPos(Pa, qa);
* break;
* case 1:
* p->next = (Link)malloc(sizeof(LNode));
* p = NextPos(P, p);
* p->data = qb->data;
* qb = NextPos(Pb, qb);
* }
* }
* while(qa){
* p->next = (Link)malloc(sizeof(LNode));
* p = NextPos(P, p);
* p->data = qa->data;
* qa = NextPos(Pa, qa);
* }
* while(qb){
* p->next = (Link)malloc(sizeof(LNode));
* p = NextPos(P, p);
* p->data = qb->data;
* qb = NextPos(Pb, qb);
* }
* p->next = NULL;
* return P;
* }
* void AddTo(Polynomial &Pa, Polynomial Pb){
* Polynomial P = AddPoly(Pa, Pb);
* DestroyList(Pa);
* Pa = P;
* }
* Polynomial MinusPoly(Polynomial P1, Polynomial P2){
* Polynomial P;
* InitPoly(P);
* CopyPoly(P, P2);
* for(Link p = NextPos(P, GetHead(P)); p; p = p->next)p->data.coef = -p->data.coef;
* AddTo(P, P1);
* return P;
* }
* Polynomial MulPoly(Polynomial Pa, Polynomial Pb){
* Polynomial P, Q;//P为临时多项式Q为返回的多项式
* InitPoly(P); InitPoly(Q);
* for(Link p = NextPos(Pb, GetHead(Pb)); p; p = p->next){
* CopyPoly(P, Pa);
* for(Link q = NextPos(P, GetHead(P)); q; q = q->next){
* q->data.coef \*= p->data.coef;
* q->data.expn += p->data.expn;
* }
* AddTo(Q, P);
* ClearList(P);
* }
* DestroyList(P);
* return Q;
* }
* Polynomial DiffPoly(Polynomial P){
* Polynomial Q;
* InitPoly(Q);
* CopyPoly(Q, P);
* Link p, q;
* for(p = GetHead(Q); p->next; ){
* if(!p->next->data.expn){
* DelFirst(p, q);
* FreeNode(q);
* continue;
* }
* p->next->data.coef \*= p->next->data.expn;
* p->next->data.expn--;
* p = p->next;
* }
* return Q;
* }
* Polynomial ExpPoly(Polynomial P, int n){
* Polynomial Q;
* InitPoly(Q);
* CopyPoly(Q, P);
* while(--n)Q = MulPoly(Q, P);//多项式求幂
* return Q;
* }
* float ValueOf(Polynomial P, float a){
* float ret = 0;
* for(Link p = NextPos(P, GetHead(P)); p; p = p->next){
* ret += p->data.coef \* pow(a, p->data.expn);//多项式在a处的值
* }
* return ret;
* }
* void IntPoly(Polynomial P){ //求原函数
* for(Link p = NextPos(P, GetHead(P)); p; p = p->next){
* p->data.expn++;
* p->data.coef /= p->data.expn;
* }
* }
* float IntOn(Polynomial P, float a, float b){
* Polynomial Q;
* InitPoly(Q);
* CopyPoly(Q, P);
* IntPoly(Q); //求积分
* float ret = ValueOf(Q, b) - ValueOf(Q, a);
* DestroyList(Q);
* return ret;
* }
* Polynomial P[N] = {NULL};
* bool spare[N];
* int main(){
* int m, x, y, z;
* float a, b;
* while(1){
* system("CLS");
* printf("这是myx做的一个还凑合能用的计算器\n");
* printf("0 ---- 结束\n");
* printf("1 ---- 创建多项式\n");
* printf("2 ---- 显示多项式\n");
* printf("3 ---- 销毁多项式\n");
* printf("4 ---- 清空多项式\n");
* printf("5 ---- 复制多项式\n");
* printf("6 ---- 多项式求和\n");
* printf("7 ---- 多项式求差\n");
* printf("8 ---- 多项式求积\n");
* printf("9 ---- 多项式求导\n");
* printf("10 --- 多项式赋值\n");
* printf("11 --- 多项式积分\n");
* printf("12 --- 多项式求幂\n");
* printf("13 --- 多项式全部显示\n");
* for(int i = 0; i < N; i++){
* if(!GetHead(P[i]))spare[i] = true;
* else spare[i] = false;
* }
* printf("当前空闲位置：");
* for(int i = 0; i < N; i++) if(spare[i])printf("%d ", i);
* printf("\n当前已用位置：");
* for(int i = 0; i < N; i++) if(!spare[i])printf("%d ", i);
* printf("\n请选择菜单:");
* scanf("%d", &m);
* switch(m){
* case 0: return 0;
* case 1:
* printf("已选择1.创建多项式\n");
* printf("请输入新建多项式的下标:");
* scanf("%d", &x);
* printf("请输入新建多项式的长度:");
* scanf("%d", &y);
* printf("请输入多项式:");
* CreatPoly(P[x], y);
* break;
* case 2:
* printf("当前已选择2.显示多项式\n");
* printf("请输入显示多项式的下标:");
* scanf("%d", &x);
* if(spare[x]){
* printf("错误!\n");
* break;
* }
* PrintPoly(P[x]);
* break;
* case 3:
* printf("当前已选择3.销毁多项式\n");
* printf("请输入销毁多项式的下标:");
* scanf("%d", &x);
* DestroyList(P[x]);
* break;
* case 4:
* printf("当前已选择4.清空多项式\n");
* printf("请输入清空多项式的下标:");
* scanf("%d", &x);
* ClearList(P[x]);
* break;
* case 5:
* printf("当前已选择5.复制多项式\n");
* printf("请输入要复制多项式的下标:");
* scanf("%d", &x);
* if(spare[x]){
* printf("错误!位置%d当前为空。\n", x);
* break;
* }
* printf("复制到:");
* scanf("%d", &y);
* if(!spare[y]){
* printf("错误!位置%d当前不可用。\n", y);
* break;
* }
* InitPoly(P[y]);
* CopyPoly(P[y], P[x]);
* break;
* case 6:
* printf("当前已选择6.多项式求和\n");
* printf("请输入被加多项式的下标:");
* scanf("%d", &x);
* if(spare[x]){
* printf("错误!位置%d当前为空。\n", x);
* break;
* }
* printf("请输入待加多项式的下标:");
* scanf("%d", &y);
* if(spare[y]){
* printf("错误!位置%d当前为空。\n", y);
* break;
* }
* printf("储存到位置:");
* scanf("%d", &z);
* if(!spare[z]){
* printf("错误!位置%d当前占用。\n", z);
* break;
* }
* P[z] = AddPoly(P[x], P[y]);
* break;
* case 7:
* printf("当前已选择7.多项式求差\n");
* printf("请输入被减多项式的下标:");
* scanf("%d", &x);
* if(spare[x]){
* printf("错误!位置%d当前为空。\n", x);
* break;
* }
* printf("请输入待减多项式的下标:");
* scanf("%d", &y);
* if(spare[y]){
* printf("错误!位置%d当前为空。\n", y);
* break;
* }
* printf("储存到位置:");
* scanf("%d", &z);
* if(!spare[z]){
* printf("错误!位置%d当前占用。\n", z);
* break;
* }
* P[z] = MinusPoly(P[x], P[y]);
* break;
* case 8:
* printf("当前已选择8.多项式求积\n");
* printf("请输入被加多项式的下标:");
* scanf("%d", &x);
* if(spare[x]){
* printf("错误!位置%d当前为空。\n", x);
* break;
* }
* printf("请输入待加多项式的下标:");
* scanf("%d", &y);
* if(spare[y]){
* printf("错误!位置%d当前为空。\n", y);
* break;
* }
* printf("储存到位置:");
* scanf("%d", &z);
* if(!spare[z]){
* printf("错误!位置%d当前占用。\n", z);
* break;
* }
* P[z] = MulPoly(P[x], P[y]);
* break;
* case 9:
* printf("当前已选择9.多项式求导\n");
* printf("请输入要求导多项式的下标:");
* scanf("%d", &x);
* if(spare[x]){
* printf("错误!位置%d当前为空。\n", x);
* break;
* }
* printf("保存到:");
* scanf("%d", &y);
* if(!spare[y]){
* printf("错误!位置%d当前不可用。\n", y);
* break;
* }
* P[y] = DiffPoly(P[x]);
* break;
* case 10:
* printf("当前已选择10.多项式赋值\n");
* printf("请输入要赋值多项式的下标:");
* scanf("%d", &x);
* if(spare[x]){
* printf("错误!位置%d当前为空。\n", x);
* break;
* }
* printf("请输入x = ");
* scanf("%f", &a);
* printf("多项式"); PrintPoly(P[x]); printf("在x = %f处的值为%f\n", a, ValueOf(P[x], a));
* break;
* case 11:
* printf("当前已选择11.多项式积分\n");
* printf("请输入要求定积分多项式的下标:");
* scanf("%d", &x);
* if(spare[x]){
* printf("错误!位置%d当前为空。\n", x);
* break;
* }
* printf("请输入积分上下限a, b:");
* scanf("%f%f", &a, &b);
* printf("多项式"); PrintPoly(P[x]); printf("在[%f, %f]上的定积分为%f\n", a, b, IntOn(P[x], a, b));
* break;
* case 12:
* printf("当前已选择12.多项式求幂\n");
* printf("请输入要求幂多项式的下标:");
* scanf("%d", &x);
* if(spare[x]){
* printf("错误!位置%d当前为空。\n", x);
* break;
* }
* printf("请输入要求的次幂");
* scanf("%d", &z);
* if(z <= 0){
* printf("错误!");
* break;
* }
* printf("保存到:");
* scanf("%d", &y);
* if(!spare[y]){
* printf("错误!位置%d当前不可用。\n", y);
* break;
* }
* P[y] = ExpPoly(P[x], z);
* break;
* case 13:
* printf("当前已选择13.多项式全部显示\n");
* for(int i=0;i<=9;i++){
* if(!spare[i]){
* PrintPoly(P[i]);printf("\n");
* }
* else printf("0\n");
* }
* break;
* }
* system("PAUSE");
* }
* return 0;
* }

1. 【运行结果】

* 
* 
* 

1. 【实验报告】

**该题目详情见大作业多项式计算器的实验报告，利用了链表之间的创建，查找，修改等操作，需要对指数部分和系数部分分别考虑，对链表的连接等处理操作要求较高。**