

**《数 据 结 构》**

**实验报告**

|  |  |
| --- | --- |
| **项目名称：** | 线性表的实验--线性表的应用 |
| **班 级：** |  |
| **学生姓名:** |  |
| **学生学号：** |  |
| **指导教师：** |  |
| **成 绩：** |  |

20 年月 日

成都信息工程大学 计算机学院

1. 实现功能描述

1.用顺序表和链表分别分别编程实现教材中例子2-1与2-2。要求：

（1） 只能用C语言编程实现；

（2） 完全 保持书中算法2.1与算法2.2形式，不允许有任何变化，除非语法上不允许;

所调用各函数参照书中19页的功能描述，其中函数名、参数个数及性质、函数功能必须与书中完全一致，不能有变化。

2.利用线性表表示一元多项式完成多项式的加、减运算。要求：

（1）输入的一元多项式可以采用只输入各项的系数与指数这种简化的方式。如对于多项式2x^2+6x^5，输入可为： 2,2 6,5 这样的简单形式。

（2）遇到有消项时应当处理，如2x^2+6x^5与3x^2-6x^5进行相加时，结果为5\*x^2。

（3）当给定x的值时，能计算表达式相加或相减的结果。

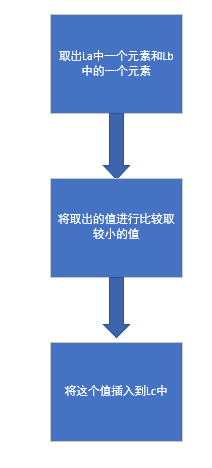
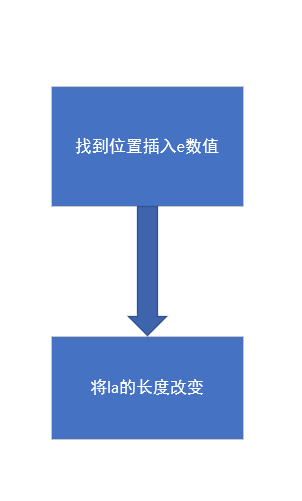
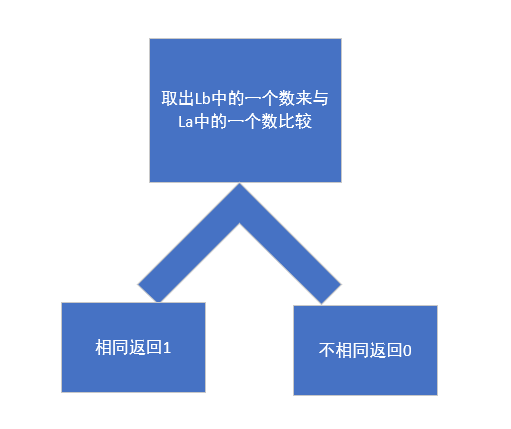
（4）操作的结果放入一个新线性表中，原来的两个表达式存储表示不变，也可以不是产生新的线性表，而是将两上线性表合并为一个。

（5）要求程序功能模块划分合理（每个函数功能单一、可重用性好），使用空间尽可能少，算法尽可能高效。

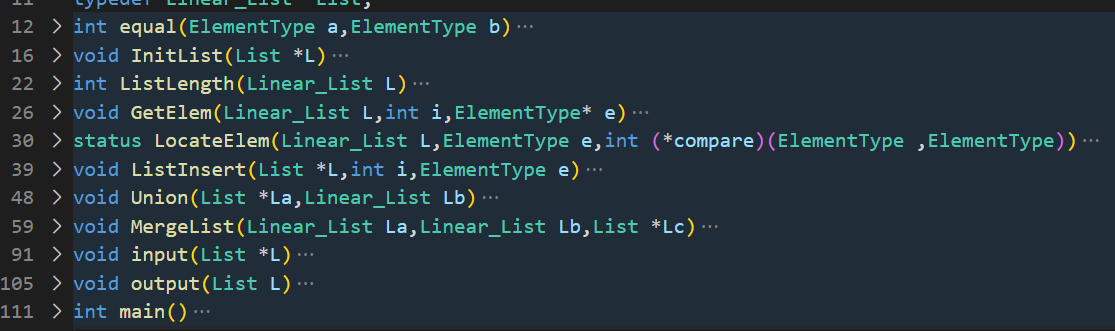
1. 方案比较与选择
   1. 从数据结构的逻辑结构与存储结构角度提供多种解决方案；
   2. 从时空效率角度分析决定最终采用方案的原因。

我用链表实现的线性表实现一元多项式的加减运算，因为省空间，最差空间复杂度为(len1+len2)，最好空间复杂度为（1），平均空间复杂度为（len1+len2）/2

1. 设计算法描述
2. 用简单示例结合所设计算法采用的数据逻辑结构图、存储结构图说明算法思想。



1. 进行模块划分，给出**主要**功能组成框图。形式如下：



1. 算法实现

**顺序表实现线性表**

**#include<stdio.h>**

**#include<string.h>**

**#include<stdlib.h>**

**typedef int status;**

**typedef int ElementType;**

**typedef struct**

**{**

**ElementType \*Data;**

**int ListLength;**

**}Linear\_List;**

**typedef Linear\_List\* List;**

**int equal(ElementType a,ElementType b)**

**{**

**return a==b;**

**}**

**void InitList(List \*L)**

**{**

**\*L=(List)malloc(sizeof(Linear\_List));**

**(\*L)->ListLength=0;**

**(\*L)->Data=NULL;**

**}**

**int ListLength(Linear\_List L)**

**{**

**return L.ListLength;**

**}**

**void GetElem(Linear\_List L,int i,ElementType\* e)**

**{**

**\*e=L.Data[i-1];**

**}**

**status LocateElem(Linear\_List L,ElementType e,bool (\*compare)(ElementType ,ElementType))**

**{**

**for(int i=0;i<L.ListLength;++i)**

**{**

**if(equal(e,L.Data[i]))**

**return i+1;**

**}**

**return 0;**

**}**

**void ListInsert(List \*L,int i,ElementType e)**

**{**

**ElementType\* newbase=(ElementType\*)realloc((\*L)->Data,((\*L)->ListLength+1)\*(sizeof(ElementType)));**

**(\*L)->Data=newbase;**

**for(int pos=(\*L)->ListLength;pos>=i;--pos)**

**(\*L)->Data[pos]=(\*L)->Data[pos-1];**

**(\*L)->Data[i-1]=e;**

**++(\*L)->ListLength;**

**}**

**void Union(List \*La,Linear\_List Lb)**

**{**

**int La\_len=ListLength(\*\*La),Lb\_len=ListLength(Lb);**

**ElementType e;**

**for(int i=1;i<=Lb\_len;++i)**

**{**

**GetElem(Lb,i,&e);**

**if(!LocateElem(\*\*La,e,equal))**

**ListInsert(La,La\_len++,e);**

**}**

**}**

**void MergeList(Linear\_List La,Linear\_List Lb,List \*Lc)**

**{**

**InitList(Lc);**

**int i=1,j=1,k=0;**

**int La\_len=ListLength(La),Lb\_len=ListLength(Lb);**

**ElementType ai,bj;**

**while(i<=La\_len && j<=Lb\_len)**

**{**

**GetElem(La,i,&ai);**

**GetElem(Lb,j,&bj);**

**if(ai<=bj)**

**{**

**ListInsert(Lc,++k,ai);**

**++i;**

**}**

**else**

**{**

**ListInsert(Lc,++k,bj);**

**++j;**

**}**

**}**

**while(i<=La\_len)**

**{**

**GetElem(La,i++,&ai);**

**ListInsert(Lc,++k,ai);**

**}**

**while(j<=Lb\_len)**

**{**

**GetElem(Lb,j++,&bj);**

**ListInsert(Lc,++k,bj);**

**}**

**}**

**void input(List \*L)**

**{**

**InitList(L);**

**printf("请输入元素个数：\n");**

**int n;**

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

**printf("请输入元素：\n");**

**ElementType e;**

**for(int i=1;i<=n;++i)**

**{**

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

**ListInsert(L,i,e);**

**}**

**}**

**void output(List L)**

**{**

**for(int i=0;i<L->ListLength;++i)**

**printf("%d ",L->Data[i]);**

**putchar('\n');**

**}**

**int main()**

**{**

**List A,B,C;**

**input(&A);**

**input(&B);**

**Union(&A,\*B);**

**// MergeList(\*A,\*B,&C);**

**output(A);**

**// output(B);**

**// output(C);**

**return 0;**

**}**

**链表实现线性表**

**#include<stdio.h>**

**#include<string.h>**

**#include<stdlib.h>**

**typedef int status;**

**typedef int ElementType;**

**typedef struct Ln**

**{**

**ElementType Data;**

**struct Ln\* next;**

**}Linear\_List;**

**typedef Linear\_List\* List;**

**bool equal(ElementType a,ElementType b)**

**{**

**return a==b;**

**}**

**void InitList(List \*L)**

**{**

**\*L=(List)malloc(sizeof(Linear\_List));**

**(\*L)->Data=0;//头结点数据表示线性表长度**

**(\*L)->next=NULL;**

**}**

**int ListLength(Linear\_List L)**

**{**

**return L.Data;**

**}**

**void GetElem(Linear\_List L,int i,ElementType\* e)**

**{**

**List p=&L;**

**while(i--)**

**p=p->next;**

**\*e=p->Data;**

**}**

**status LocateElem(Linear\_List L,ElementType e,bool (\*compare)(ElementType ,ElementType))**

**{**

**List p=&L;**

**for(int i=1;p!=NULL;++i,p=p->next)**

**{**

**if(equal(e,p->Data))**

**return i;**

**}**

**return 0;**

**}**

**void ListInsert(List \*L,int i,ElementType e)**

**{**

**List newbase=(List)malloc(sizeof(Linear\_List));**

**newbase->Data=e;**

**++(\*L)->Data;**

**List p=(\*L);**

**while(--i)**

**p=p->next;**

**newbase->next=p->next;**

**p->next=newbase;**

**}**

**void Union(List \*La,Linear\_List Lb)**

**{**

**int La\_len=ListLength(\*\*La),Lb\_len=ListLength(Lb);**

**ElementType e;**

**for(int i=1;i<=Lb\_len;++i)**

**{**

**GetElem(Lb,i,&e);**

**if(!LocateElem(\*\*La,e,equal))**

**ListInsert(La,La\_len++,e);**

**}**

**}**

**void MergeList(Linear\_List La,Linear\_List Lb,List \*Lc)**

**{**

**InitList(Lc);**

**int i=1,j=1,k=0;**

**int La\_len=ListLength(La),Lb\_len=ListLength(Lb);**

**ElementType ai,bj;**

**while(i<=La\_len && j<=Lb\_len)**

**{**

**GetElem(La,i,&ai);**

**GetElem(Lb,j,&bj);**

**if(ai<=bj)**

**{**

**ListInsert(Lc,++k,ai);**

**++i;**

**}**

**else**

**{**

**ListInsert(Lc,++k,bj);**

**++j;**

**}**

**}**

**while(i<=La\_len)**

**{**

**GetElem(La,i++,&ai);**

**ListInsert(Lc,++k,ai);**

**}**

**while(j<=Lb\_len)**

**{**

**GetElem(Lb,j++,&bj);**

**ListInsert(Lc,++k,bj);**

**}**

**}**

**void input(List \*L)**

**{**

**InitList(L);**

**printf("请输入元素个数：\n");**

**int n;**

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

**printf("请输入元素：\n");**

**ElementType e;**

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

**{**

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

**ListInsert(L,i,e);**

**}**

**}**

**void output(List L)**

**{**

**for(List p=L->next;p!=NULL;p=p->next)**

**printf("%d ",p->Data);**

**putchar('\n');**

**}**

**int main()**

**{**

**List A,B,C;**

**input(&A);**

**input(&B);**

**// Union(&A,\*B);**

**MergeList(\*A,\*B,&C);**

**// output(A);**

**// output(B);**

**output(C);**

**return 0;**

**}**

**一元多项式加减**

**#include<stdio.h>**

**#include<string.h>**

**#include<stdlib.h>**

**#include<math.h>**

**typedef int status;**

**typedef int ElementType;**

**typedef struct Ln**

**{**

**ElementType coefficient;**

**ElementType exponent;**

**struct Ln\* next;**

**}Linear\_List;**

**typedef Linear\_List\* List;**

**bool equal(ElementType a,ElementType b)**

**{**

**return a==b;**

**}**

**void InitList(List \*L)**

**{**

**\*L=(List)malloc(sizeof(Linear\_List));**

**(\*L)->coefficient=0;//头结点数据表示线性表长度**

**(\*L)->exponent=0;**

**(\*L)->next=NULL;**

**}**

**int ListLength(Linear\_List L)**

**{**

**return L.coefficient;**

**}**

**void GetElem(Linear\_List L,int i,ElementType\* c,ElementType\* e)**

**{**

**List p=&L;**

**while(i--)**

**p=p->next;**

**\*c=p->coefficient;**

**\*e=p->exponent;**

**}**

**void ListInsert(List \*L,int i,ElementType c,ElementType e)**

**{**

**List newbase=(List)malloc(sizeof(Linear\_List));**

**newbase->coefficient=c;**

**newbase->exponent=e;**

**++(\*L)->coefficient;**

**List p=(\*L);**

**while(--i)**

**p=p->next;**

**newbase->next=p->next;**

**p->next=newbase;**

**}**

**void MergeList(Linear\_List La,Linear\_List Lb,List \*Lc,int op)**

**{**

**InitList(Lc);**

**int i=1,j=1,k=0;**

**int La\_len=ListLength(La),Lb\_len=ListLength(Lb);**

**ElementType ac,ae,bc,be;**

**while(i<=La\_len && j<=Lb\_len)**

**{**

**GetElem(La,i,&ac,&ae);**

**GetElem(Lb,j,&bc,&be);**

**if(ae<be)**

**{**

**ListInsert(Lc,++k,ac,ae);**

**++i;**

**}**

**else if(ae>be)**

**{**

**ListInsert(Lc,++k,op\*bc,be);**

**++j;**

**}**

**else**

**{**

**if(ac+op\*bc!=0)**

**ListInsert(Lc,++k,ac+op\*bc,ae);**

**++i,++j;**

**}**

**}**

**while(i<=La\_len)**

**{**

**GetElem(La,i++,&ac,&ae);**

**ListInsert(Lc,++k,ac,ae);**

**}**

**while(j<=Lb\_len)**

**{**

**GetElem(Lb,j++,&bc,&be);**

**ListInsert(Lc,++k,op\*bc,be);**

**}**

**}**

**void input(List \*L)**

**{**

**InitList(L);**

**printf("请输入单项式个数：\n");**

**int n;**

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

**printf("请按照指数递增输入单项式：\n");**

**ElementType c,e;**

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

**{**

**scanf("%d %d",&c,&e);**

**ListInsert(L,i,c,e);**

**}**

**}**

**void output(List L)**

**{**

**for(List p=L->next;p!=NULL;p=p->next)**

**printf("%d %d,",p->coefficient,p->exponent);**

**putchar('\n');**

**}**

**ElementType calculation(List L,ElementType x)**

**{**

**ElementType ans=0;**

**for(List p=L->next;p!=NULL;p=p->next)**

**ans+=p->coefficient\*pow(x,p->exponent);**

**return ans;**

**}**

**int main()**

**{**

**List A,B,C;**

**int op;**

**ElementType x;**

**printf("请输入第一个一元多项式的系数和指数\n");**

**input(&A);**

**printf("请输入第二个一元多项式的系数和指数\n");**

**input(&B);**

**printf("请输入运算符，1表示+，-1表示-\n");**

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

**MergeList(\*A,\*B,&C,op);**

**// output(A);**

**// output(B);**

**output(C);**

**printf("请给定x一个值：\n");**

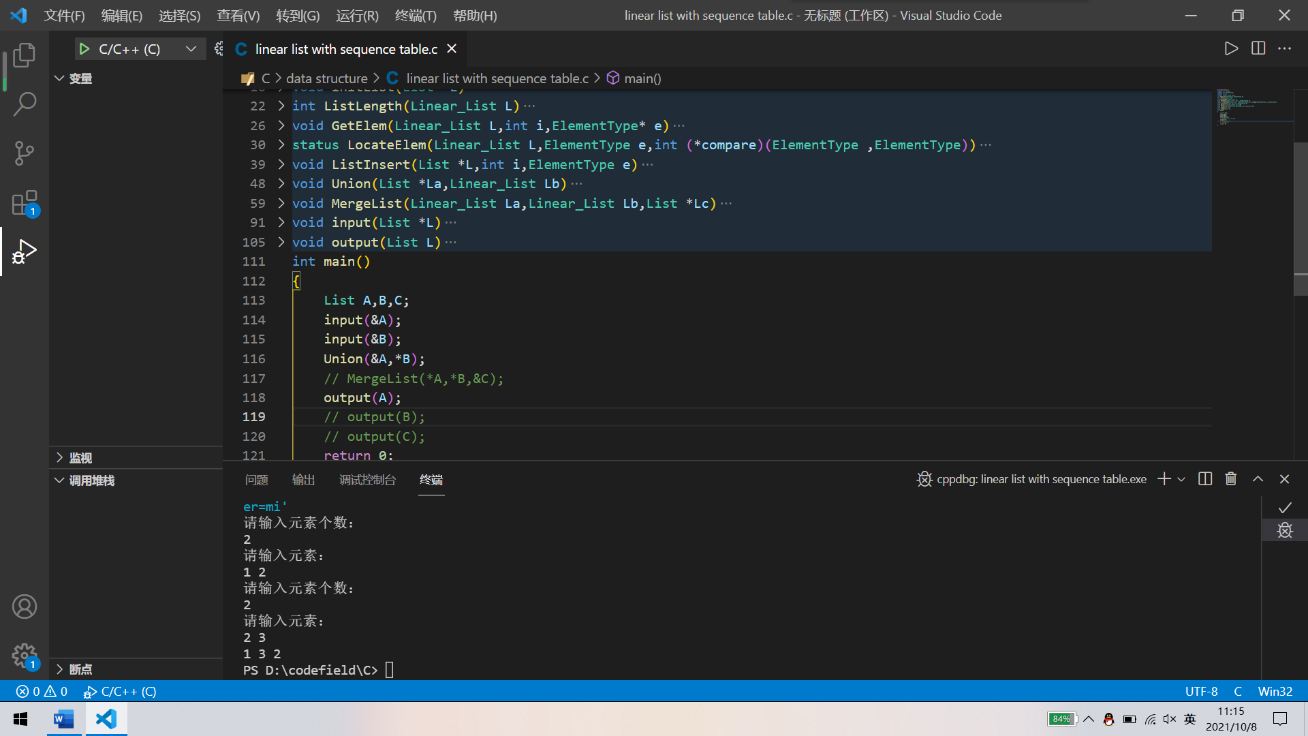
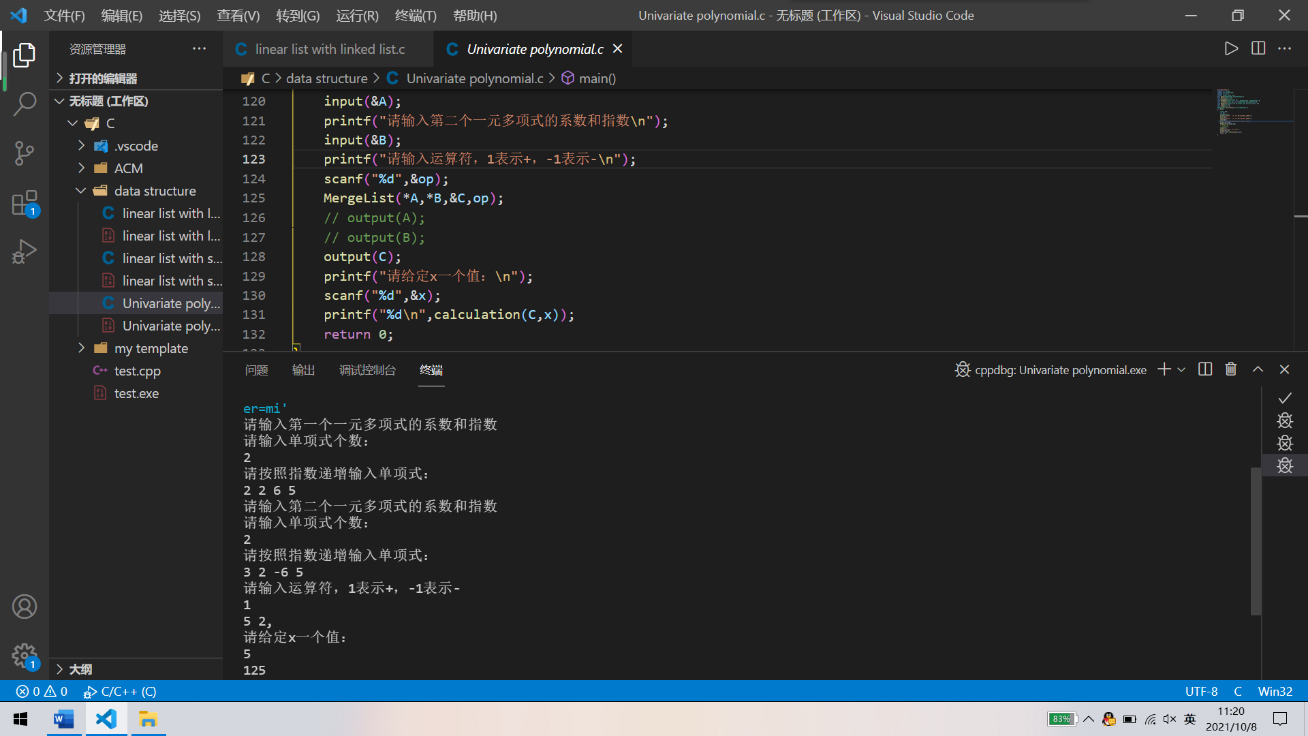
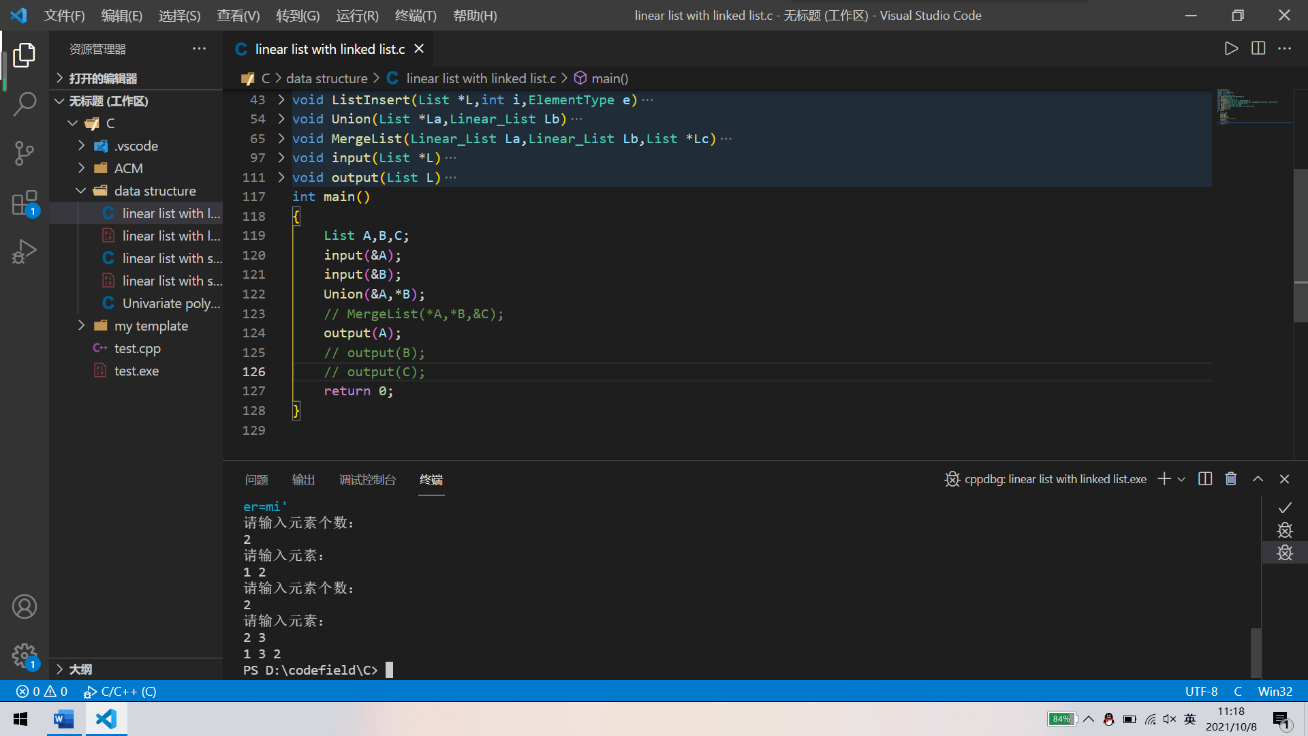
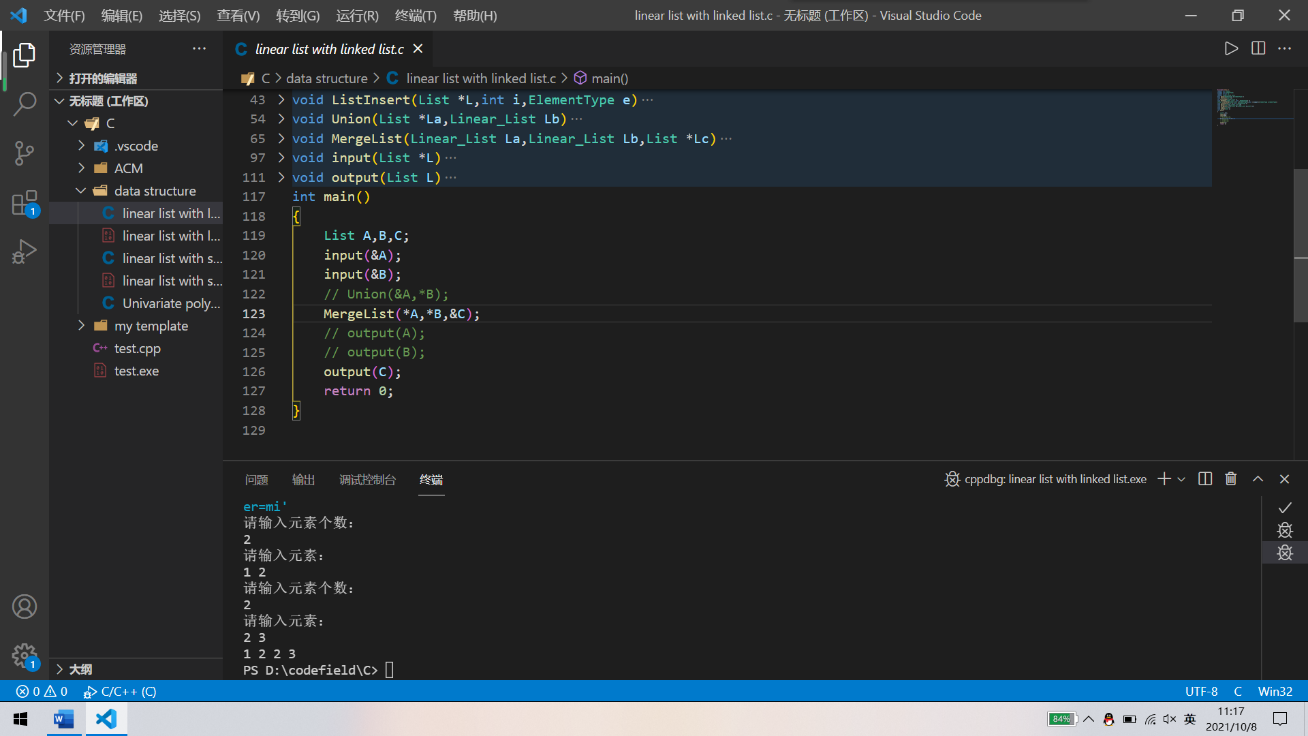
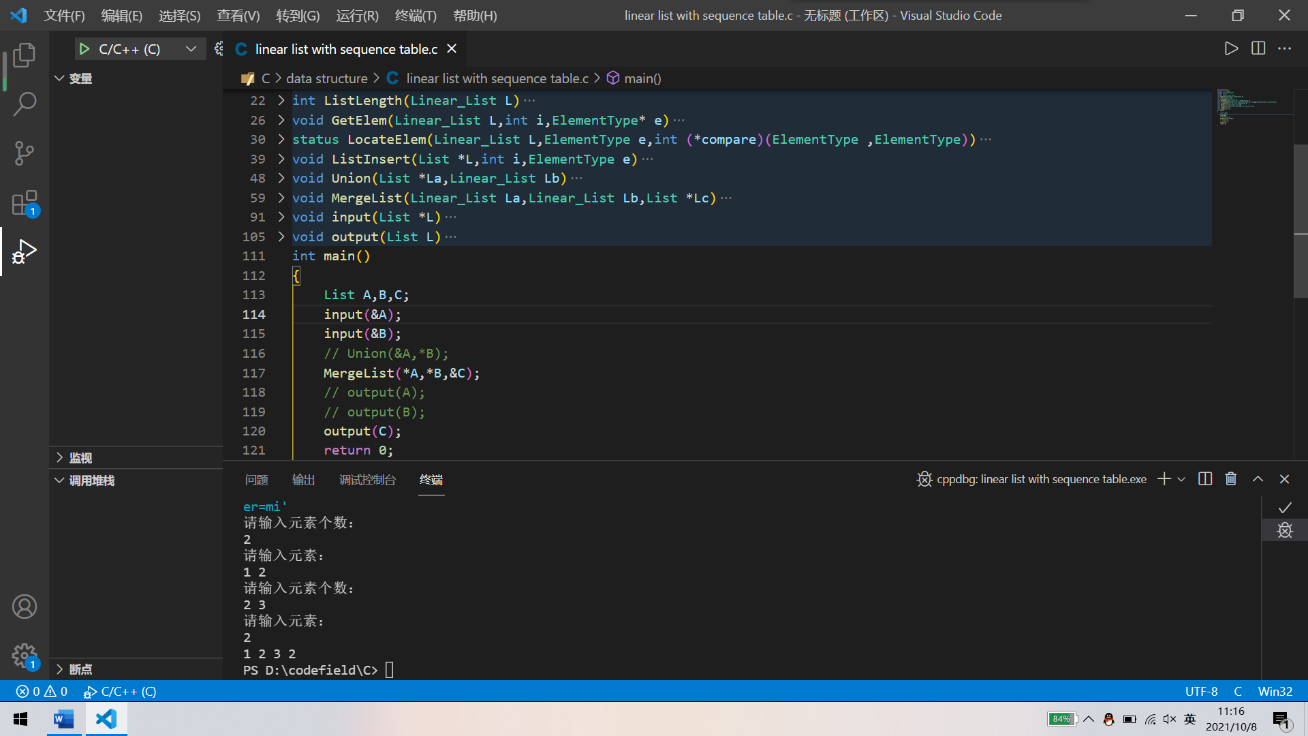
**scanf("%d",&x);**

**printf("%d\n",calculation(C,x));**

**return 0;**

**}**

1. 实验结果测试与分析
2. 用各种可能数据测试程序，取截图；



1. 对结果进行分析，说明算法的有效性。

均符合手动运算结果

1. 思考及学习心得
2. 描述实验过程中对此部分知识的认识；
3. 特别描述在学习方法上的收获及体会；
4. 针对前面的思考题内容**选择部分**在此回答。

线性表是数据结构入门，丑陋实现很简单，但是要考虑内存问题就复杂多了，时间不够，随便实现就完事了

大一下写过类似的，当时花了1天（好像），现在1个晚上就写完了（熟能生巧）

相减同理相加，值得注意的是，在相减中要注意两个指数相同时，要注意他们两个系数相减是否为0，如果为0则不进行存储，反之建立新的结点，进行存储相减之后的系数和指数。最后输入x的值进行最后的运算并输出结果。