# 实验二 顺序表的建立及基本操作方法实现

实验预备知识：

1．熟练运用指针进行程序设计，掌握结构体指针。

2．掌握使用结构体指针访问结构体变量。

3．掌握指针作为函数的参数使用。

4．理解单链表的含义、目的和处理方法。

## 一、实验目的

1．掌握线性表的链式存贮结构及基本操作，深入了解链表的基本特性，以便在实际问题背景下灵活运用它们。

2．巩固该存贮结构的构造方法，深入理解和灵活掌握链表的插入、删除等操作。

## 二、实验要求（本次实验要求上交）

【题目1】----验证型

实现循环单链表的各种基本运算的算法。参考代码见（3-1.cpp）

该程序主函数完成如下功能：

1. 初始化循环单链表
2. 利用尾插法插入若干元素
3. 输出循环单链表
4. 输出该单链表的长度
5. 判断该单链表是否为空
6. 输出单链表的第3个元素
7. 在地4的元素位置上插入元素f
8. 删除第3个元素
9. 销毁该循环单链表

【题目2】

编写一个程序，用单链表存储一元多项式，将单链表按幂指数的降序排序并实现两个多项式的相加运算。源程序见附录部分。

该工程的程序结构图如下图所示：

CreatListR

Add

DestroyList

Sort

DispPolyy

main

测试数据：

A:{1.2，0}{2.5，1}{3.2，3}{-2.5，5}

B: {-1.2，0}{2.5，1}{3.2，3}{2.5，5}{5.4,10}

|  |
| --- |
| 算法分析： |
| 问题1：单链表排序  将头结点的下一个节点赋给指针p，如果p不为空，把p结点的下一个节点的地址赋给第三个同类型的结构体变量r，将p的下一个指针域置为空，然后让p节点后移一位，然后进入循环，当p不为空的时候，将p的下一节点赋给变量r，然后当q(此时是头结点)的下一个节点不为空的时候与头结点的下一个节点的指数值比p节点(此时p是第三个节点)的指数值大的时候，q结点后移一位，再与p节点比较，当不满足循环条件的时候，判断q节点的下一节点是否与p节点相等，若相等就不做任何改变，否则将p,q交换位置，最后再将r赋给交换后的p节点，重复第一个循环(当p不等于空的时候),完成排序。  问题2：利用多项式进行相加运算  将待相加的一个多项式ha的首地址赋给最终的多项式hc，当ha的下一个节点不为空的时候，ha后移一位，移动结束后ha为该多项式的最后一个节点，然后令ha指向另一个多项式hb的第二个节点的地址(因为这两个多项式的头结点都没有存储多项式的值)，这样就完成了两个多项式的相加，到时候输出就从最终的多项式头结点的下一个节点进行输出 |

请自己编写Add（）函数及main（）函数，其余函数已经给出（见3-2.cpp）。将整个程序的源码附录于下表中。

|  |
| --- |
| 请将源程序附录于此： |
| #include <stdio.h>  #include <malloc.h>  #define MAX 20 //多项式最多项数  typedef struct //定义存放多项式的数组类型  {  double coef; //系数  int exp; //指数  }PolyArray;  typedef struct pnode //定义单链表结点类型  {  double coef; //系数  int exp; //指数  struct pnode \*next;  }PolyNode;  void DispPoly(PolyNode \*L) //输出多项式  {  bool first=true; //first为true表示是第一项  PolyNode \*p=L->next;  while (p!=NULL)  {  if (first)  first=false;  else if (p->coef>0)  printf("+");  if (p->exp==0)  printf("%g",p->coef);  else if (p->exp==1)  printf("%gx",p->coef);  else  printf("%gx^%d",p->coef,p->exp);  p=p->next;  }  printf("\n");  }  void DestroyList(PolyNode \*&L) //销毁单链表  {  PolyNode \*p=L,\*q=p->next;  while (q!=NULL)  {  free(p);  p=q;  q=p->next;  }  free(p);  }  void CreateListR(PolyNode \*&L,PolyArray a[],int n) //尾插法建表  {  PolyNode \*s,\*r;int i;  L=(PolyNode \*)malloc(sizeof(PolyNode)); //创建头结点  L->next=NULL;  r=L; //r始终指向终端结点,开始时指向头结点  for (i=0;i<n;i++)  {  s=(PolyNode \*)malloc(sizeof(PolyNode));//创建新结点  s->coef=a[i].coef;  s->exp=a[i].exp;  r->next=s; //将\*s插入\*r之后  r=s;  }  r->next=NULL; //终端结点next域置为NULL  }  void Sort(PolyNode \*&head) //按exp域递减排序  {  PolyNode \*p=head->next,\*q,\*r;  if (p!=NULL) //若原单链表中有一个或以上的数据结点  {  r=p->next; //r保存\*p结点后继结点的指针  p->next=NULL; //构造只含一个数据结点的有序表  p=r;  while (p!=NULL)  {  r=p->next; //r保存\*p结点后继结点的指针  q=head;  while (q->next!=NULL && q->next->exp>p->exp)  q=q->next; //在有序表中找插入\*p的前驱结点\*q  if(q->next==p) ;  else{  p->next=q->next; //将\*p插入到\*q之后  q->next=p;  }  p=r;  }  }  }  void Add(PolyNode \*&ha,PolyNode \*hb,PolyNode \*&hc) //求两有序集合的并  {  hc=ha;  while (ha->next!=NULL)  {  ha=ha->next;  }  ha->next=hb->next;  }  void main()  {  PolyArray polynome1[4],polynome2[5];  PolyNode \*L,\*R,\*S;  int i;  for(i=0;i<4;i++){  printf("请输入系数:");  scanf("%f",&polynome1[i].coef);  printf("请输入指数:");  scanf("%f",&polynome1[i].exp);  }  CreateListR(L,polynome1,4);  Sort(L);  DispPoly(L);  for(i=0;i<5;i++){  printf("请输入系数:");  scanf("%f",&polynome2[i].coef);  printf("请输入指数:");  scanf("%f",&polynome2[i].exp);  }  CreateListR(R,polynome2,5);  Sort(R);  DispPoly(R);  Add(L,R,S);  Sort(S);  DispPoly(S);  DestroyList(L);  DestroyList(R);  DestroyList(S);  } |
| 请将运行结果图附录于此 |
| C:\Users\Administrator\AppData\Roaming\Tencent\Users\1061892125\QQ\WinTemp\RichOle\DC3(1}XTZ_OEQJ5E6}(U1KH.png |

## 实验上传方法：

将源程序文件和本word文档（添加了相应的源程序和截图）上传。多个文件请将其放入一个文件夹压缩后上传。请于指定时间前上交，过时不补！