《数据结构》实验报告1

班级:DL062048 姓 名:白顺文 学 号:2020303092 E-mail:2662269845@qq.com日期:2021/3/28

**◎实验题目: 合并有序数组**

**◎实验目的：将两个有序数组合并成一个有序数组并输出**

**◎实验内容：使用c语言将上述实验目的实现**

**一、需求分析**

说明程序设计的任务，强调的是程序要做什么，明确规定：

1. 输入形式：

第一行为第一个有序数组的长度，正整数n, n<=20;

第二行为第一个有序数组的n个数字，用空格隔开;

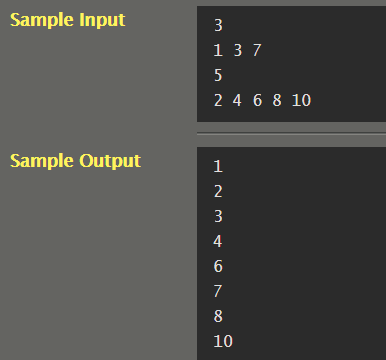
第三行为第二个有序数组的长度，正整数m,m<=20;

第四行为第二个有序数组的m个数字，用空格隔开；

2、输出形式：输出合并后的数组，每个数字占一行；

3、实现功能：给定两个按照升序排列的有序数组，请把它们合成一个升序数组并输出。

4、测试数据：如图给出测试数据，若存在相同项，也则重复输出



补充：

1.将两个有序数组有序合并，只有一个主函数，定义出三个整数型数组a,b,c，引入整数变量m,n,s,i,j,k，第一行数组个数输进n，第二行数组元素输入a，第三行数组个数输进m，第四行数组个数输进b，将a，b中的数组元素最后整合排序进c数组，先比较a，b中数组元素大小，升序排序时，小的元素先进入c数组，当其中一个数组输入完，则把另一个未输入完的数组直接续上。

**(二) 概要设计**

算法：合并有序数组，比较数组大小，将两数组依次输入第三数组。为了实现上述操作，应定义两个20上限的数组和一个40上限的合并数组。

1．本程序包含一个模块：

主程序模块：

main()

{

Input a;

Input b;

Compare a and b;

Input c;

Output c;

}

**(三) 详细设计**

#include <stdlib.h>

#include <stdio.h>

int main()

{

int i, j = 0, k=0, m, n,s;

int a[20], b[20], c[40];

scanf("%d", &n);

for (i = 0; i < n; i++) {

scanf("%d", &a[i]);

}

scanf("%d", &m);

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

scanf("%d", &b[i]);

}

i=0;

while(i<n&&j<m)

{

if(a[i]<=b[j])//如果a小则a先进

{

c[k]=a[i];

i++;k++;//然后a进入下一个元素

}

else

{

c[k]=b[j] //否则b先进;

j++;k++;//b进入下一个元素

}

}

while(i<n)//将未续完的一个数组续完

{

c[k]=a[i];

k++;i++;

}

while(j<m)

{

c[k]=b[j];

k++;j++;

}

s=m+n;

for(i=0;i<s;i++) printf("%d\n",c[i]);//输出数组元素

return 0;

}

**(四) 程序使用说明及测试结果**

1．程序使用说明

（1）本程序的运行环境为VC6.0。

（2）输入数据：

第一行为第一个有序数组的长度，正整数n, n<=20;

第二行为第一个有序数组的n个数字，用空格隔开;

第三行为第二个有序数组的长度，正整数m,m<=20;

第四行为第二个有序数组的m个数字，用空格隔开；

2．测试结果：n＝3，3个数组元素依次为：1，3,7，m值为5， 5个数组元素依次为：2,4,6,8,10

则这正确的结果输出顺序为：

1

2

3

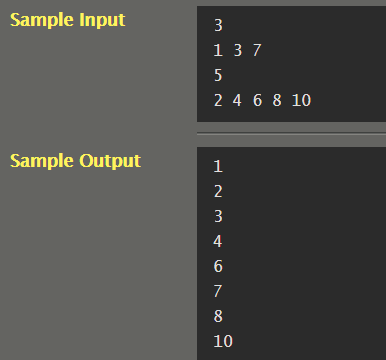
4

6

7

8

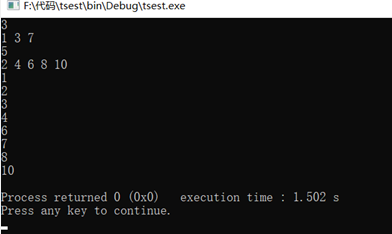
10



3.调试过程中遇到的问题是如何解决提以及对设计与实现的回顾讨论和分析：

出现了数组输入过程中未考虑数组下标而引起了越界问题，经过修正后已正常运行。

4. 运行界面：



**（五）、实验总结(实验心得)**

   这个题比较基础，只是考察一个比较大小排序输入的算法，不过要注意输入数组元素时考虑数组个数和数组下标的问题，保证数组输入输出不越界，在设置数组大小的时候看清题目要求，最后输出数组大小建议为40。整体难度不高。

**教师评语：**

**实验成绩：**

指导教师签名：

批阅日期：

《数据结构》实验报告2

班级:DL062048 姓 名:白顺文 学 号:2020303092 E-mail:2662269845@qq.com日期:2021/4/2

**◎实验题目: 高精度计算Pi值**

**◎实验目的：通过反三角函数公式计算Pi的值，任意指定位数并输出；**

**◎实验内容：利用双向循环链表实现通过反三角函数公式计算出Pi的值，并按输出数据输出相应位数的π值。**

**一、需求分析**

描述：限制使用双向链表作存储结构，请根据用户输入的一个整数（该整数表示精确到小数点后的位数，可能要求精确到小数点后500位），高精度计算PI值。可以利用反三角函数幂级展开式来进行计算；



输入：输入的一个正整数n；

输出：输出PI的值，精确到小数点后n位，最后输出一个回车。

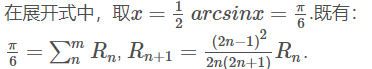
样例：输入样例

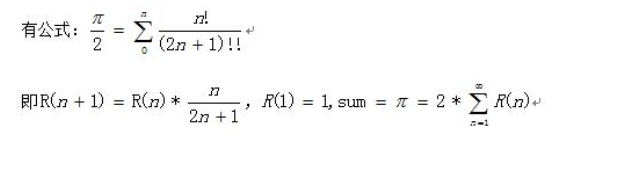
5

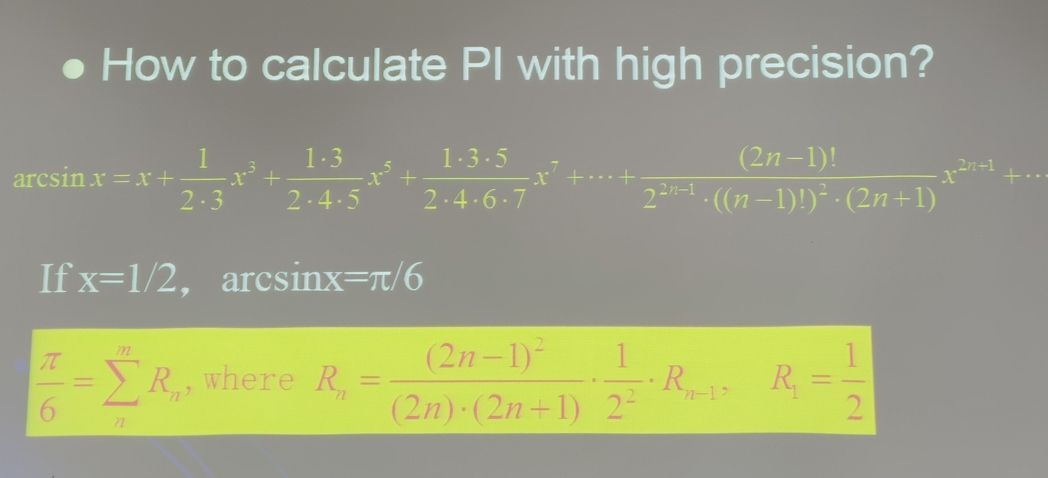
输出样例

3.14159

提示用反三角函数的幂级展开式来进行计算。







**二 概要设计**

**先创建四个函数，分别是**

**LinkCreate():创建链表；**

**Sum():链表加法；**

**Mul();链表乘法进位；**

**Div():链表除法；**

**main():主函数进行递进加法实现反三角函数的公式计算。**

**主函数中调用前面四个函数实现链表数据之间的加乘除运算，每个节点存储每一位数据，最后根据输入的位数输出小数点后n位圆周率。**

有效件头文件为<stdio.h>

以此实现手算公式流程。

**三 详细设计**

typedef struct Node：创建双链表节点

void Create(LinkList L);

{

头插法建立双链表，包含头结点，节点共有max个

}

void Sum(LinkList a,LinkList b)

{

LinkList p=a->pre,q=b->pre;

int x;

while(q没到b节点){

x=p和q的和;

q->data=个位上数;

q->pre->data+=x的十位上的数;

q节点往前;

p节点往前;

}

}

void Mul(LinkList a,int n);

{

LinkList p=a->pre;

int x,y=0;

for(p没到a节点，就一直往前虚)

{

x=p乘n再加上上一个节点的进位;

y=x的十位数;

p->data=x的个位数;

}

x=(p->data)\*n+y;

y=x/10;

p->data=x%10;

}

void Div(LinkList a,int n)

{

LinkList p=a->next;

int x,y=0;

for(p没到a就一直往后续)

{

x=p加上上一个节点个位数余位;

p->data=x除以n后数的十位上的数;

y= x除以n后数的个位上的数;

}

}

main()

{

int n,i,top,bot,s=1;

输入n；

创建sum和rad双链表；

LinkCreate（sum和rad）

赋R1，将公式中左边π/6的分母乘过去，首项为1/2\*6=3；

sum->next->data=3;

rad->next->data=3;

while(计算1000次，保证精度)

{

top= (2\*s-1)\*(2\*s-1);公式分子

bot=8\*s\*(2\*s+1);公式分母

Multiply(rad,top);将链表rad乘分子

Divide(rad,bot);再将链表rad乘分母

Sum(rad,sum);rad和sum相加构造Rn+Rn+1，并将其存在sum中，rad为计算下一个循环中的Rn+2保留

s++;递进

}

if(输入为0) 那么输出“3”;

else {

sum=sum->next;

printf("%d.",sum->data);其实是输出“3.”

for(i=0;i<n;i++){

printf("%d",sum->next->data);

sum=sum->next;将链表中的东西输出出来，循环输出n位

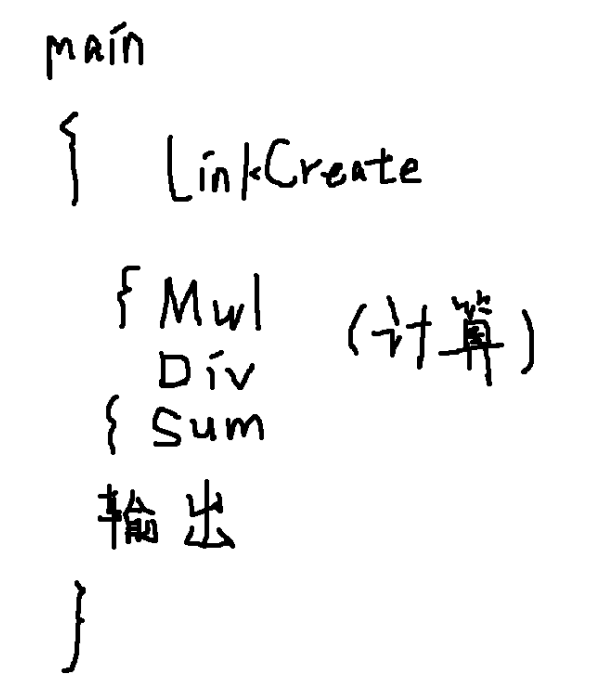
}

}

printf("\n");

return 0;

}

**ps：源代码为**

#define \_CRT\_SECURE\_NO\_DEPRECATE

#include <stdio.h>

#include <stdlib.h>

#include <iostream>

#define MAX 1000

typedef struct Node

{

int data;

struct Node\* pre;

struct Node\* next;

}LNode, \* LinkList;

void LinkCreate(LinkList L) //初始化双循环链表，置每个节点data为0

{

LinkList p = L, q;

int i;

L->next = L->pre = L;

for (i = 0; i < MAX; i++)

{

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

q->data = 0;

p->next = q;

q->pre = p;

q->next = L;

L->pre = q;

p = q;

}

}

void Sum(LinkList a, LinkList b) //加法，从后往前pre

{

LinkList p = a->pre, q = b->pre;

int x;

while (q != b) {

x = q->data + p->data;

q->data = x % 10;

q->pre->data += x / 10;

q = q->pre;

p = p->pre;

}

}

void Mul(LinkList a, int n) //乘法，从后往前pre

{

LinkList p = a->pre;

int x, y = 0;

for (; p != a; p = p->pre)

{

x = (p->data) \* n + y;

y = x / 10;

p->data = x % 10;

}

x = (p->data) \* n + y;

y = x / 10;

p->data = x % 10;

}

void Div(LinkList a, int n) //除法，从前往后next

{

LinkList p = a->next;

int x, y = 0;

for (; p != a; p = p->next)

{

x = p->data + y \* 10;

p->data = x / n;

y = x % n;

}

}

int main()

{

int n, i, top, bot, s = 1;

scanf("%d", &n);

LinkList sum, rad;

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

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

LinkCreate(sum);

LinkCreate(rad);

sum->next->data = 3;//首项

rad->next->data = 3;//首项

while (s < 2000)

{

top = (2 \* s - 1) \* (2 \* s - 1);

bot = 8 \* s \* (2 \* s + 1);

Mul(rad, top);

Div(rad, bot);

Sum(rad, sum);

s++;

}

if (n == 0) printf("3\n");

else {

sum = sum->next;

printf("%d.", sum->data);

for (i = 0; i < n; i++) {

printf("%d", sum->next->data);

sum = sum->next;

}

}

printf("\n");

return 0;

}

（从vs中复制过来）

**四 使用说明、测试分析及结果**

1. **输入要圆周率要精确的位数，然后输出相应精度的圆周率。**
2. **输入测试数据**

5

**输出结果为**

3.14159

**3、调试过程中发现对于头结点是否进行乘除都可以得到正确结果，多次调试验证；**

**4、运行界面如下：**



**五、实验总结**

本次计算pi利用的只是反三角函数的公式，通过递进公式，第一项为3，不断递进，无穷小项不断相加，使得精度更高，而要注意的是题目要求的精度至少500，为保证精度我在程序中相加循环次数设置为999次。

本题难度在于要正确设计出加乘除函数，考虑到运算法则中的进位余位问题，真正模拟出手算流程，然后正常运算并将其输出。输出时要考虑n为0的情况，直接输出3而非3.。

要将公式中的π单独放在一边，直接使得π的值被得到并且输出。

本次题目我参考了我搜得的一些资料，包括以往做noj中大数乘法加法的经验，并多次调试，检验正确输出，和正确数据对比。

本次题目中我最大收获是理解链表间的计算，实现出模拟手算的效果。

**教师评语：**

**实验成绩：**

指导教师签名：

批阅日期：