《数据结构》实验报告

班 级: DL062138 姓 名: 张苏宇 学 号: 2021302853

E-mail:905159071@qq.com 日 期: 2022/3/20

◎实验题目:T001合并有序数组

◎实验目的:将两个有序数组合并成一个新的有序数组并输出

◎实验内容:使用c语言将上述实验目的实现

**一、需求分析**

1. 输入的形式和输入值的范围

第一行输入第一个有序数组的长度n, n<=20；

第二行输入第一个有序数组的n个数字，用空格隔开；

第三行输入第二个有序数组的长度m，m<=20;

第四行输入第二个有序数组的m个数组，用空格隔开。

2. 输出的形式

输出合并后的数组，每个数字占一行

3. 程序所能达到的功能

（1）定义三个数组和两个变量

（2）输入数据

（3）合并前两个数组到第三个数组

（4）输出合并后的数组，程序结束

4. 测试数据：

正确的输入输出：

输入：3

1 3 7

5

2 4 6 8 10

输出：1

2

3

4

6

7

8

10

**二 概要设计**

算法：合并有序数组，比较数组大小，将两个数组依次输入第三个数组。为了实现上述操作，应定义两个上限为20的数组和一个上限为40的合并数组。

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 = 0, j = 0, k = 0, l = 0, m, n;

int a[20] = {0}, b[20] = {0}, c[40] = {0};

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]);

}

//输入a，b两个数组

while (l < n || j < m) {

if (l == n)

c[k++] = b[j++];

else if (j == m)

c[k++] = a[l++];

//判断读取a，b两数组是否结束，若结束，将剩余未读取元素放入新数组

else if (a[l] <= b[j])

c[k++] = a[l++];

//如果a小则a先进，然后a进入下一个元素

else

c[k++] = b[j++];

//否则b先进，b进入下一个元素

}

int s = m + n;

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

printf("%d\n", c[i]);

return 0;

}

本次实验中数据结构和算法设计

1. 数据结构：用了顺序表----数组，可以通过下标来索引数组中的元素。
2. 算法：算法我利用了归并排序的思想，可以把两个有序的序列合并为一个大的有序数列。

**四 使用说明、测试分析及结果**

1. **说明如何使用你编写的程序；**

编译运行后输入第一个序列元素个数 n ，回车，然后输入第一个序列的n个数，回车。然后再输入第二个序列元素个数m，回车，输入第二个序列的m个数，回车。

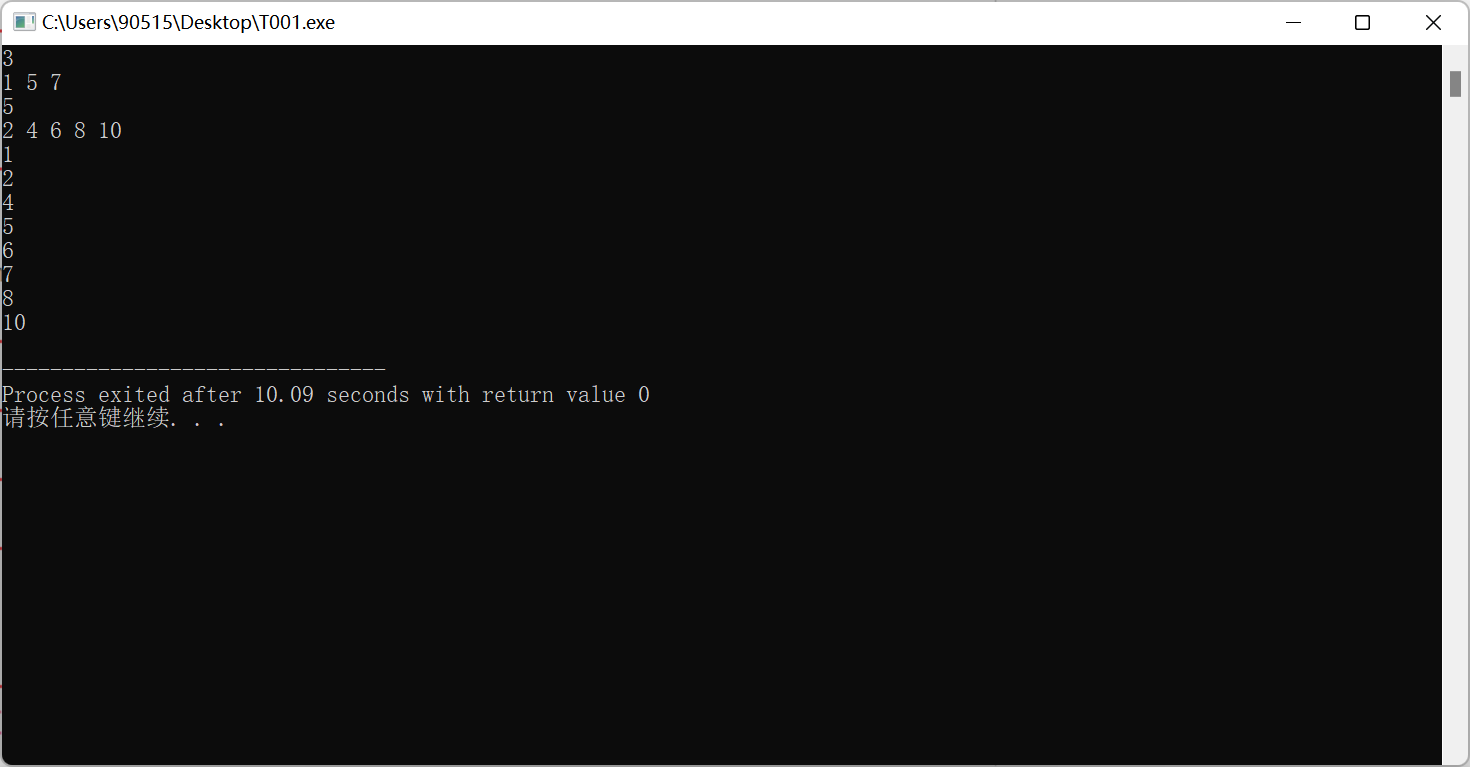
1. **测试结果与分析；**

当输入n = 3 ，m = 5，两个序列分别为 2 4 5 和 2 6 8 10 11 时，程序输出的结果为 2 2 4 5 6 8 10 11 ，易得结果正确。

1. **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**

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。

1. **运行界面**



1. **实验总结**

时间复杂度：O(m+n)。

数组移动单调递增，最多移动 m+n 次，因此时间复杂度为 O(m+n)。

空间复杂度：O(m+n)。

需要建立长度为 m+n 的中间数组。

此题目较简单，考察了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。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

班 级: DL062138 姓 名: 张苏宇 学 号: 2021302853

E-mail:905159071@qq.com 日 期: 2022/3/20

◎实验题目: T002高精度计算Π值

◎实验目的：通过反三角函数公式计算Π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Π的值，并按输出数据输出相应位数的Π值。

**一、需求分析**

1. 输入的形式和输入值的范围

一个正整数n，代表精确到小数点后多少位 ，n <= 500

2. 输出的形式

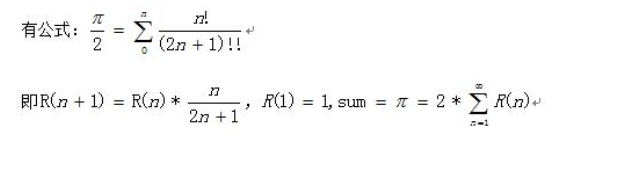
输出Π的值，精确到小数点后n位，最后输出一个回车。

1. 程序所能达到的功能
2. 设计链表节点
3. 读入n的值
4. 调用各种模块计算Π
5. 输出Π
6. 测试数据：

输入：20

输出：3.14159265358979323846

**二 概要设计**



1. 本程序包括以下模块
2. 输入数据模块
3. 主程序模块
4. 创建链表节点和初始化链表模块
5. 循环计算pi值模块
6. 输出链表模块
7. 模块调用图

主程序模块 -> 创建链表和初始化链表模块 -> 输入数据模块 -> 循环计算pi值模块 -> 输出链表模块

LinkCreate():头插法建立双链表，包含头结点，节点共有max个；

Sum():两个链表相加，从后往前p = p->pre

Mul():链表乘法，对链表每一位 \*n，从后往前p = p->pre

Div():链表除法，对链表每一位 /n，从前往后p = p->next

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。

**三 详细设计**

#include <stdio.h>

#include <stdlib.h>

#define MAX 1000 // 设定求小数点后多少位

typedef struct Node { // 双向链表的结构体

int data;

struct Node \*pre;

struct Node \*next;

} LNODE, \* LinkList;

void LinkCreate(LinkList L) ; //初始化双循环链表，并且将每个节点data设为0

void Sum(LinkList a, LinkList b) ; //两个链表相加，从后往前p = p->pre

void Mul(LinkList a, int n) ; //乘法，对链表每一位 \*n，从后往前p = p->pre

void Div(LinkList a, int n) ; //除法，对链表每一位 /n，从前往后p = p->next

int main() {

int n, up, down, s = 0;

scanf("%d", &n);

LinkList sum, temp;

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

temp = (LinkList)malloc(sizeof(LNODE));

//创建两个链表的头结点

LinkCreate(sum);

LinkCreate(temp);

//初始化双循环链表

sum->next->data = 2;//首项为2

temp->next->data = 2;//首项为2

while (s++ < 2000) {//运用递推关系式，计算Π的值

up = s;

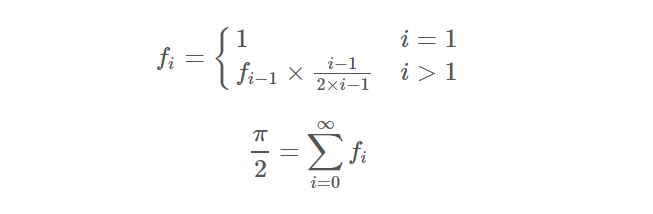
down = (2 \* s + 1);

Mul(temp, up);

Div(temp, down);

Sum(temp, sum);

}



sum = sum->next;

if (n == 0)//考虑无小数的情况

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

else {

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

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

//将链表中的数值输出出来，循环输出n位

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

sum = sum->next;

}

printf("\n");

//其实我更想用puts（）;

}

return 0;

}

void LinkCreate(LinkList L) { //初始化双循环链表，并且将每个节点data设为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) { //两个链表相加，从后往前p = p->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，对链表每一位 \*n，从后往前p = p->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，对链表每一位 /n，从前往后p = p->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;

}

}

**乘法的设计：**乘法我们模拟竖式的乘法运算，考虑到这是一个 *大数乘法* ，因此我们只需要从尾部到头部依次对每一位做乘法即可。

目前大数乘法算法主要有以下几种思路：

模拟小学乘法：最简单的乘法竖式手算的累加型；

分治乘法：最简单的是Karatsuba乘法，一般化以后有Toom-Cook乘法；

快速傅里叶变换FFT：（为了避免精度问题，可以改用快速数论变换FNTT），时间复杂度O(N lgN lglgN)。

中国剩余定理：把每个数分解到一些互素的模上，然后每个同余方程对应乘起来就行；

Furer’s algorithm：在渐进意义上FNTT还快的算法。

**这里我采取了第一种，有良好的容错率。**先做完乘法，然后再回到尾部，再从尾到头依次处理进位，这样的好处是这两种操作分隔开了，操作起来难度不大，也比较好想。

**除法的设计：**除法也是模拟竖式运算，这个是从头到尾进行处理，这一位的数据做完除法，然后余数作为下一位的“进位”，下一位继续做除法，如果有除不尽的情况，就一直在尾部插入新的节点**直到“进位”为0，或者达到最大位数为止。**

**加法的设计：**加法设计和乘法有些类似，也是从尾部到头部处理，然后一直进位。

**四 使用说明、测试分析及结果**

1. **说明如何使用你编写的程序；**

输入要圆周率要精确的位数，然后输出相应精度的圆周率

1. **测试结果与分析；**

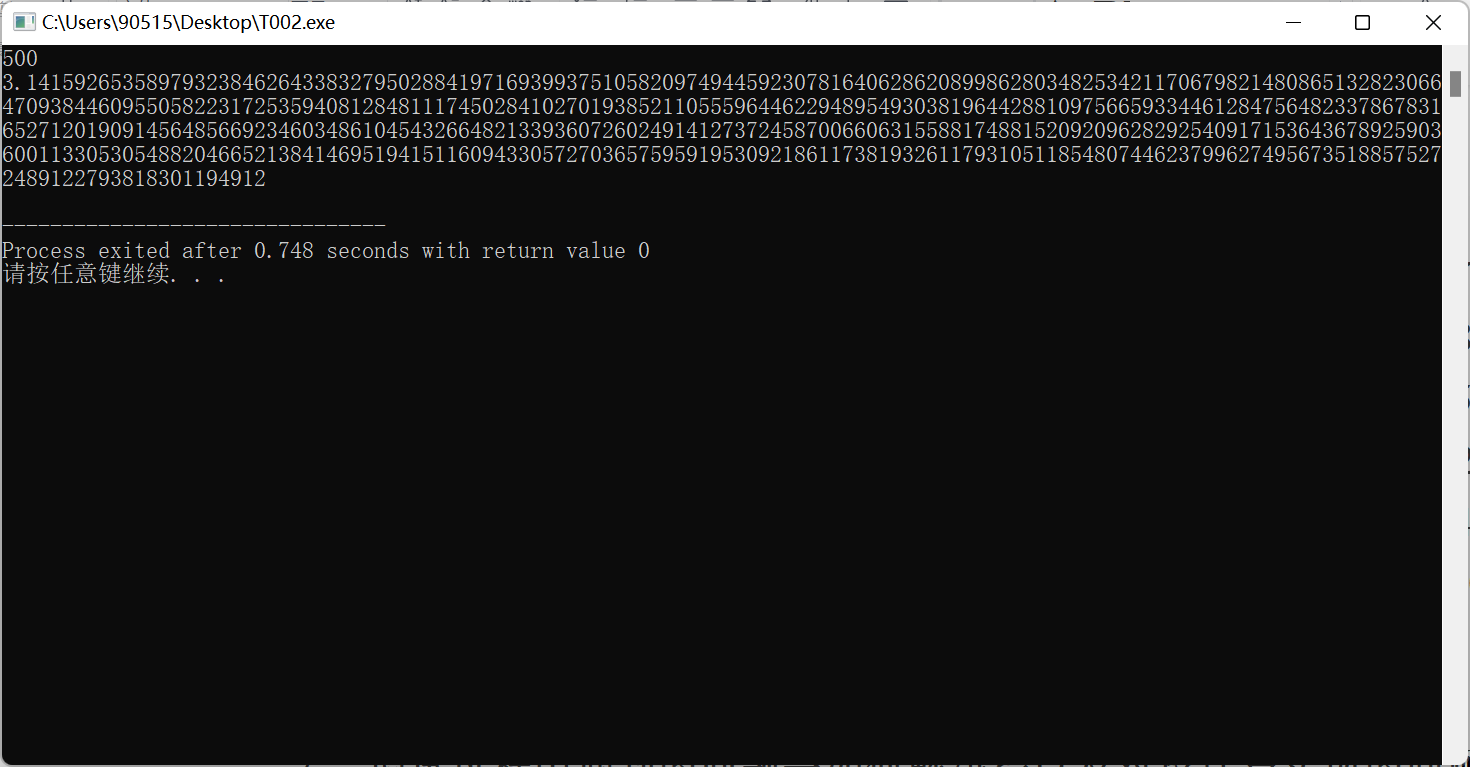
输入：500

输出：3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706798214808651328230664709384460955058223172535940812848111745028410270193852110555964462294895493038196442881097566593344612847564823378678316527120190914564856692346034861045432664821339360726024914127372458700660631558817488152092096282925409171536436789259036001133053054882046652138414695194151160943305727036575959195309218611738193261179310511854807446237996274956735188575272489122793818301194912

1. **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**

对于数学递推公式运算的顺序问题和考虑无小数位的情况

1. **运行界面**



**五、实验总结**

这一题高精度计算Π值利用的是反三角函数的公式，通过Taylor Formula（泰勒展开公式），第一项为2，展开后无穷小项不断叠加，使得精度更高。本次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2000次。

这道题的关键也就是难度在于大数加乘除的函数，考虑到运算法则中的进位余位问题，要正确设计函数，使函数有较强的健壮性。而设计函数的关键在于如何真正地用计算机模拟手算，这就要求我认真思考手算的具体步骤和主要原理。在编写本道题目的代码时我参考了搜索的一些资料，包括但不限于以往做的noj中大数乘法加法的经验，并经过多次调试检验，最终正确输出。

有一个小技巧是将公式中的π单独放在一边，直接使得π的值被得到并且输出，如果一开始算π/2的话，最后还需一个大数乘法，这样则增加了时间和空间的复杂度。其次，输出时要考虑n为0的情况，直接输出3而非3.。

**教师评语：**

**实验成绩：**

指导教师签名：

批阅日期：