

《数据结构》实验报告1

班级: 10012006

姓名: 夏卓

学号: 2020303245

E-mail: 2769223717@qq.com

日期: 2022.3.19

1.1 合并有序数组

实验内容:

File Name: T001.cpp

实验1.1: 合并有序数组

Time Limit: 3000ms , Memory Limit: 10000KB , Accepted: 0 , Total Submissions: 0

VIDEO1

Description

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。

Input

第一行为第一个有序数组的长度，正整数 n ， $n \leq 20$ ；
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 ， $m \leq 20$ ；
第四行为第二个有序数组的 m 个数字，用空格隔开。

Output

输出合并后的数组，每个数字占一行。

Sample Input

```
3
1 3 7
5
2 4 6 8 10
```

Sample Output

```
1
2
3
4
6
7
8
10
```

一、需求分析：

1. 输入：

第一行为第一个有序数组的长度，正整数 n ， $n \leq 20$ ；

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 ， $m \leq 20$ ；

第四行为第二个有序数组的 m 个数字，用空格隔开。

2. 输出：

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：

将两个升序排列的有序数组合并成一个升序排列的有序数组。

二、概要设计：

核心思想：

- 另开辟一个大小为两个输入数组长度之和的数组用于存放最终结果
- 利用双指针算法，使 $c[cnt] = \min(a[i], b[j])$ ；
- 注意需要将 a 、 b 数组中剩余未比较的元素也复制到 c 数组中

三、详细设计：

核心算法的伪码框架：

```
1 //数据结构（数组）
2 int a[N], b[N], c[2 * N];
3
4 //主函数
5 int main()
6 {
7     Input();
8     int cnt = 0, i, j; //cnt为当前数组c拷贝的数组下标，i、j分别为a、b中正在比较
                        //的数组元素下标
9
10    for (i = 0, j = 0; i < n && j < m;)
11    {
12        if (a[i] < b[j])
13            c[cnt++] = a[i++];
14        else
15            c[cnt++] = b[j++];
16    } //双指针算法使得c[cnt]=min(a[i],b[j])
17
18    while (i < n)
19        c[cnt++] = a[i++];
20    while (j < m)
21        c[cnt++] = b[j++]; //将a、b数组中剩余未比较的元素也复制到c数组中
22
23    Output();
```

```
24     return 0;
25 }
```

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

本程序的运行环境为visual studio 2019。

输入：

第一行为第一个有序数组的长度，正整数 n ， $n \leq 20$ ；

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 ， $m \leq 20$ ；

第四行为第二个有序数组的 m 个数字，用空格隔开。

最后输入回车。

程序会输出将这两个升序排列的有序数组合并成一个新的升序排列的有序数组，每个数组元素占一行

2. 测试结果与分析

本程序较好的实现了实验需求，如要满足对更大数组的合并，可适当调整全局变量 N

3. 调试过程中遇到的问题及解决方法

无

4. 运行界面



```
Microsoft Visual Studio 调试控制台
3
1 3 7
5
2 4 6 8 10
1
2
3
4
6
7
8
10
D:\Code\VS_code\homework\合并有序数组\Debug\合并有序数组.exe (进程 13112) 已退出，代码为 0。
按任意键关闭此窗口。...
```

五、实验总结

- 本实验我在编程中用时10分钟
- 调试花费了2分钟
- 本实验较为简单，可另开辟一个数组用于存储最终结果
- 主要运用了双指针算法

1.2 高精度计算PI值

实验内容：

File Name: T002.cpp

实验1.2：高精度计算PI值

Time Limit: 3000ms , Memory Limit: 10000KB , Accepted: 0 , Total Submissions: 0

VIDEO1

Description

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n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。

Input

输入的一个正整数n

Output

输出PI的值，精确到小数点后n位，最后输出一个回车。

Sample Input	5
Sample Output	3.14159

© 2002-2012 JDSoft.

一、需求分析：

1. 输入：

输入一个正整数 n ， n 可能为500

2. 输出：

输出PI的值，精确到小数点后n位，最后输出一个回车。

3. 程序所能达到的功能：

通过数学公式的级数收敛逼近PI值，至少能精确计算出PI小数点后500位

二、概要设计：

核心思想：

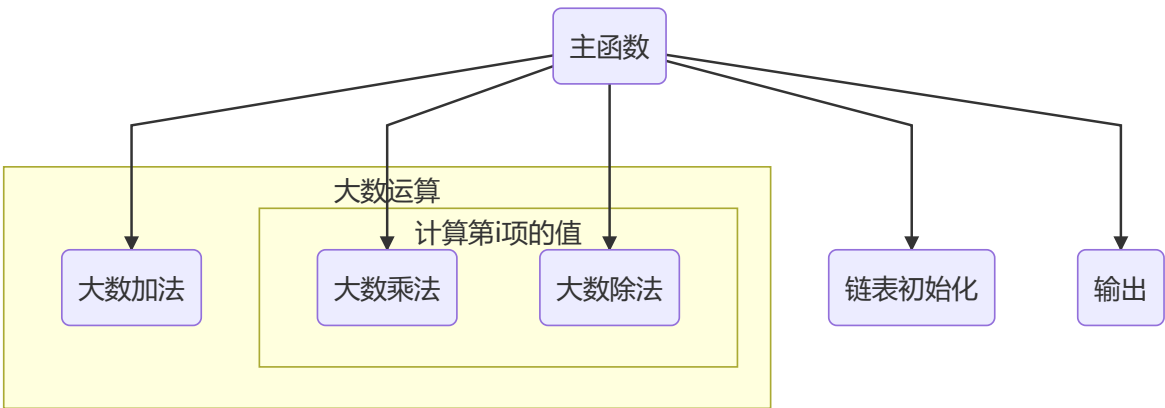
- 由于利用反三角函数幂级展开式收敛速度较慢，因此本程序利用公式 $\frac{\pi}{2} = \sum_{k=0}^{\infty} \frac{k!}{(2k+1)!!}$ 来逼近PI值
- 本程序利用两条双向循环链表 Li , Ls 作为存储单元，分别用于存储和计算求和项 $R_{(k)} = \frac{k!}{(2k+1)!!}$ 及计算结果 sum
- 通过编写函数来实现链表间的相加，以及链表所储存大数与整型数的乘，除运算。

程序框架：

本程序包含四个模块：

1. 主程序模块；
2. 初始化链表模块；
3. 支持大数运算的函数模块；
4. 输出模块

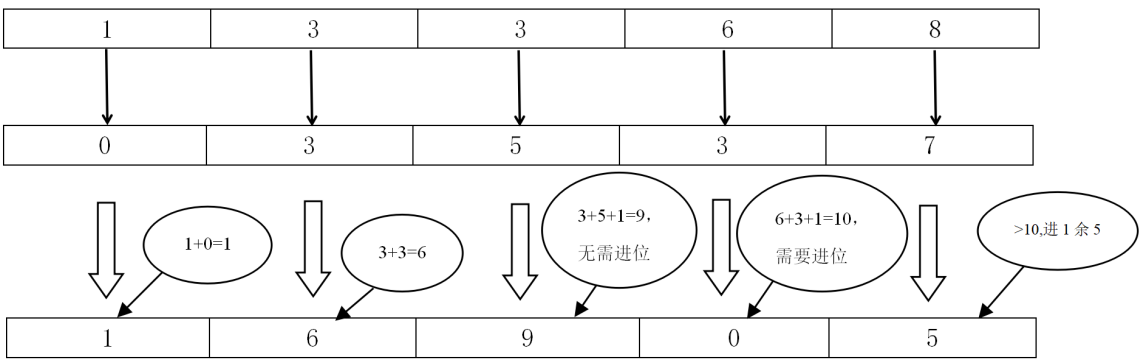
模块调用图：



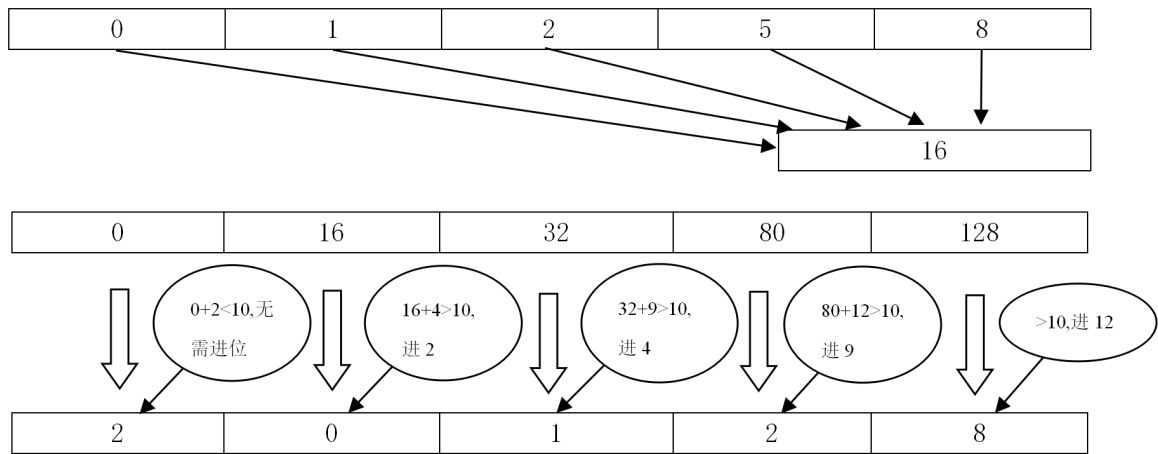
三、详细设计：

算法实现图：

(以大数加法为例)



(以大数乘法为例)



核心算法的伪码框架:

```
1 //结构体（双向循环链表）
2 struct Node
3 {
4     int data;
5     Node* pre, * next;
6 };
7
8 //主函数
9 int main()
10 {
11     Input();    //输入数据
12     Init();     //初始化链表
13     for (int i = 1; i < M; i++)    //将级数的前M项作为PI的近似值，M根据所需精度调
    整
14     {
15         CountR_i(i);    //计算R(i)的值
16         Add();    //计算前i项之和
17     }
18     Output();    //输出结果
19     return 0;
20 }
21
22 //核心算法
23 void Add()    //大数加法
24 {
25     int carry = 0;    //进位
26     for (p:Ls->pre, q:Li->pre)    //从低位到高位循环
27     {
28         carry += p->data + q->data;
29         p->data = carry % 10;
30         carry /= 10;
31     }
32 }
33
34 void Mul(int i)    //大数乘法
35 {
36     int rem = 0;    //进位
37     for (p:Ls->pre)    //从低位到高位循环
```

```

38     {
39         rem += p->data * i;
40         p->data = rem % 10;
41         rem /= 10;
42     }
43 }
44
45 void Dev(int i)    //大数除法
46 {
47     int leave = 0;    //余数
48     for (p:Ls->next)    //从高位到低位循环
49     {
50         leave = leave * 10 + p->data;
51         p->data = leave / i;
52         leave %= i;
53     }
54 }

```

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

本程序的运行环境为visual studio 2019。

输入一个整数 n ($n \geq 1$)，作为需要输出的小数点后PI的位数，接着输入回车。

程序会输出PI的值，其精确到小数点后 n 位（注意：计算结果未进行四舍五入，而是直接截断）

2. 测试结果与分析

本程序较好的实现了实验需求，如对位数有更高的要求，可适当调整全局变量 N ， M ，其中， N 表示链表长度， M 表示取逼近级数的前 M 项和作为PI的近似值

3. 调试过程中遇到的问题及解决方法

- 选取适当的数学公式对PI进行逼近，要求收敛速度尽可能快且易于用计算机进行编程

选用 $\pi = 2 \times \sum_{k=0}^{\infty} \frac{k!}{(2k+1)!!}$ 进行逼近

- 选取适当的储存结构存储大数

选用双向循环链表储存 $R(n)$ 和前 k 项和，易于进行大数加减乘除的运算

- 对大数进行加减乘除的四则运算

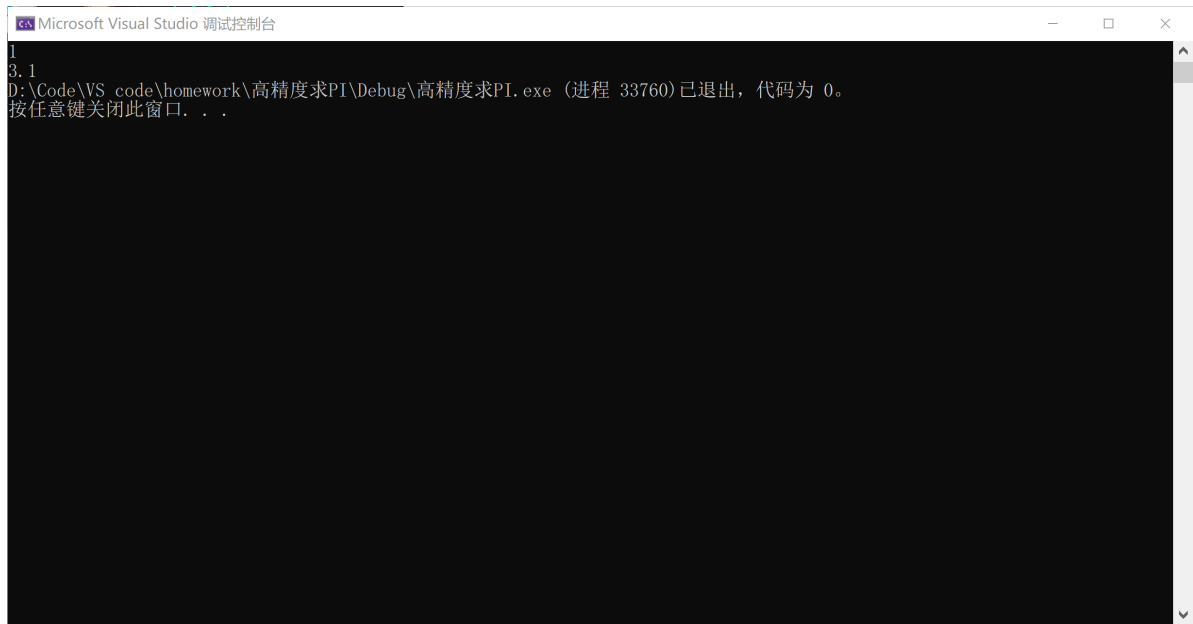
本程序采用模拟竖式手算的方法进行大数之间的四则运算，因为计算结果最高位不会进位，因此无需考虑最高位进位的问题

- 能够估计出运算次数与答案精度之间的关系

采用计算收敛级数前5000项的和来满足精度要求，链表长度取1000来满足位数要求

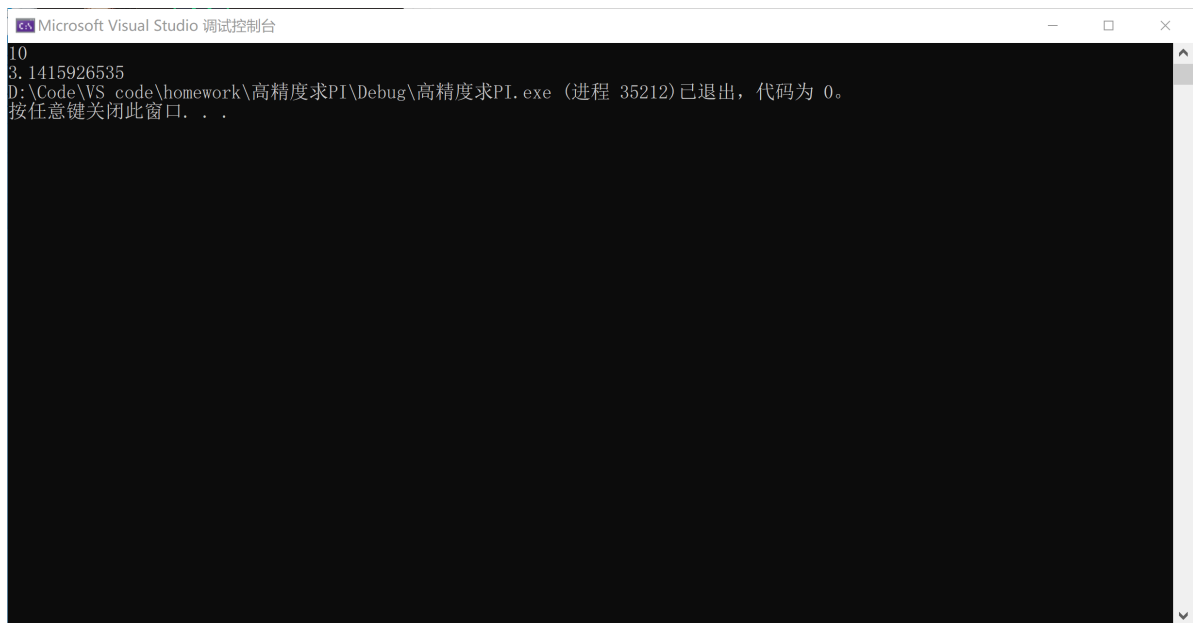
4.运行界面

正确输入1：



```
Microsoft Visual Studio 调试控制台
3.1415926535
D:\Code\VS_code\homework\高精度求PI\Debug\高精度求PI.exe (进程 33760) 已退出，代码为 0。
按任意键关闭此窗口。...
```

正确输入2：



```
Microsoft Visual Studio 调试控制台
3.1415926535
D:\Code\VS_code\homework\高精度求PI\Debug\高精度求PI.exe (进程 35212) 已退出，代码为 0。
按任意键关闭此窗口。...
```

正确输入3：


```
Microsoft Visual Studio 调试控制台
500
3. 1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421170679821480865132823066
470938446095505822317253594081284811174502841027019385211055596446229489549303819644288109756659334461284756482337867831
652712019091456485669234603486104543266482133936072602491412737245870066063155881748815209209628292540917153643678925903
600113305305488204665213841469519415116094330572703657595919530921861173819326117931051185480744623799627495673518857527
2489122793818301194912
D:\Code\VS_code\homework\高精度求PI\Debug\高精度求PI.exe (进程 30380) 已退出，代码为 0。
按任意键关闭此窗口。 . . .
```

错误输入：

```
Microsoft Visual Studio 调试控制台
0
3.
D:\Code\VS_code\homework\高精度求PI\Debug\高精度求PI.exe (进程 31048) 已退出，代码为 0。
按任意键关闭此窗口。 . . .
```

五、实验总结

- 本实验我在编程中用时50分钟
- 调试花费了10分钟，主要在初始化链表以及大数四则运算的过程中
- 一开始我想用 \arctan 的级数来对 $\frac{\pi}{4}$ 进行逼近，但该公式收敛速度太慢，预计需要循环300000次以上，因此我改用了 $\pi = 2 \times \sum_{k=0}^{\infty} \frac{k!}{(2k+1)!!}$ 进行逼近
- 在模拟大数的竖式四则运算时，我先在草稿纸上进行模拟，再予以编程实现，主要是需要注意进位的问题