

数据结构课程设计

**2022/2023(1)**



课设题目大整数运算

学生姓名 赵嵩然

学生学号 202103151528

学生班级 软工03

任课教师 毛国红

提交日期 2023.1.8

**计算机科学与技术学院**

目录

**一、实验题目和要求 1**

**二、实验开发环境 1**

三、实验课题分析 18

3.1 系统功能详细设计 18

3.2 数据存储设计 20

四、实验调试测试及分析 25

4.1 调试测试情况 21

4.2 总体分析 22

**五、实验总结 22**

参考文献（算法及相关实现如有参考，请列出） 49

图目录

图3‑1 系统功能结构图 13

图4‑1 定时任务流程图 19

图4‑2 数据库E-R实体及联系图 21

图5‑1 基础信息数据 25

图5‑2 日线行情数据文件列表 26

图5‑3 部分上交所指数信息展示 27

表目录

[表3-1 基本面数据展示功能需求 15](#_Toc5725)

[表3-2 每日数据展示功能需求 16](#_Toc19452)

[表3-3 策略选股数据展示功能需求 16](#_Toc17141)

[表4-1 股票列表ts\_stock\_basic 22](#_Toc27514)

**题目 实验报告**

备注：题目后面以图书管理系统代替，各项内容均以相关联内容示范，撰写报告时请删除备注，并将蓝色字体改成黑色。

实验报告排版要求：

1. **要求采用WORD进行文档撰写和排版；**
2. **要求有封面，目录和页码，要求采用多级标题方式进行排版；**
3. **正文字体建议为宋体，正文字体大小为小四或五号；正文行间距建议为1.5倍行间距；**
4. **正文不要用黑体或彩色；粘贴的代码可以比正文字体小一号，代码及其注释要适当排版；**
5. **实验报告中的图和表，要求进行编号。图和图的说明不要跨页，表格也尽量不要跨页；**

**要求注意排版，排版的优劣将直接影响评分；**

1. **实验题目和要求**

**【问题描述】**密码学分为两类密码：对称密码和非对称密码。对称密码主要用于数据的加/解密，而非对称密码则主要用于认证、数字签名等场合。非对称密码在加密和解密时，是把加密的数据当作一个大的正整数来处理，这样就涉及到大整数的加、减、乘、除和指数运算等，同时，还需要对大整数进行输出。请采用相应的数据结构实现大整数的加、减、乘、除和指数运算，以及大整数的输入和输出。

**【基本要求】**

1. 要求采用链表来实现大整数的存储和运算，**不允许使用标准模板类的链表类(list)和函数**。同时要求可以从键盘输入大整数，也可以文件输入大整数，大整数可以输出至显示器，也可以输出至文件。大整数的存储、运算和显示，可以同时支持二进制和十进制，但至少要支持十进制。大整数输出显示时，必须能清楚地表达出整数的位数。测试时，各种情况都需要测试，并附上测试截图；要求测试例子要比较详尽，各种极限情况也要考虑到，测试的输出信息要详细易懂，表明各个功能的执行正确；
2. 要求大整数的长度可以不受限制，即大整数的十进制位数不受限制，可以为十几位的整数，也可以为500多位的整数，甚至更长；大整数的运算和显示时，只需要考虑**正的大整数**。如果可能的话，请以秒为单位显示每次大整数运算的时间；
3. 要求采用类的设计思路，不允许出现类以外的函数定义，但允许友元函数。主函数中只能出现类的成员函数的调用，不允许出现对其它函数的调用。
4. 要求采用多文件方式：.h文件存储类的声明，.cpp文件存储类的实现，主函数main存储在另外一个单独的cpp文件中。如果采用类模板，则类的声明和实现都放在.h文件中。
5. 不强制要求采用类模板，也不要求采用可视化窗口；要求源程序中有相应注释；
6. 建议采用Visual C++ 6.0及以上版本进行调试；
7. **实验开发环境**

硬件环境：(备注：可以查看“计算机”属性)

处理器：Intel(R) Core(TM) i7-10750HCPU @ 2.4GHz

内存：16.00GB

系统类型：64位操作系统

1. **实验课题分析（主要的模块功能、流程图、算法原理、代码实现）**

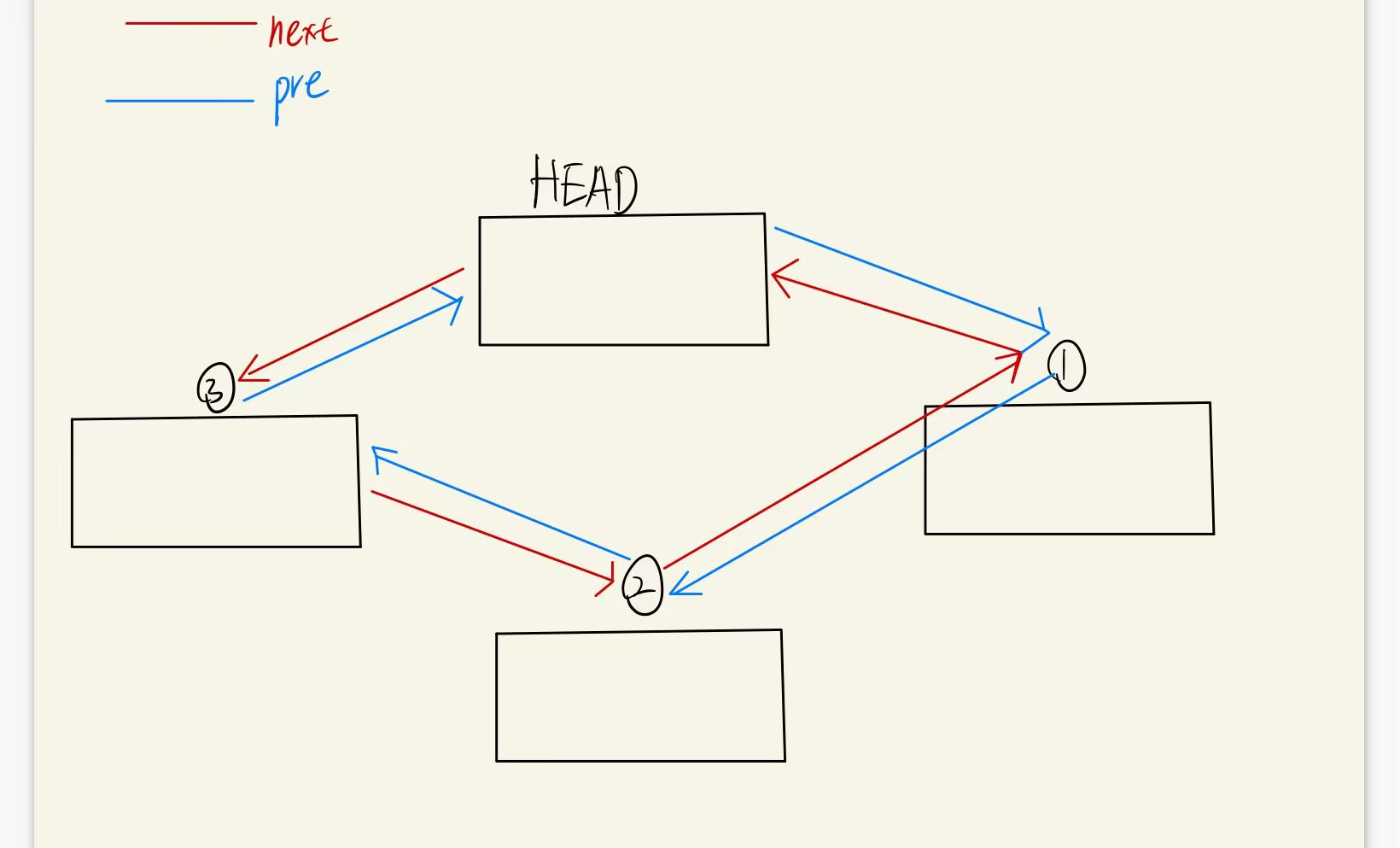
3.1类的设计

这里设计了两个类，分别是节点类和双向链表类。并且在这里把所有的数据定义为Public，函数在类外设计即可。链表的每个节点通过10000进制的方式储存输入的大整数。接下来介绍一下链表的主要功能

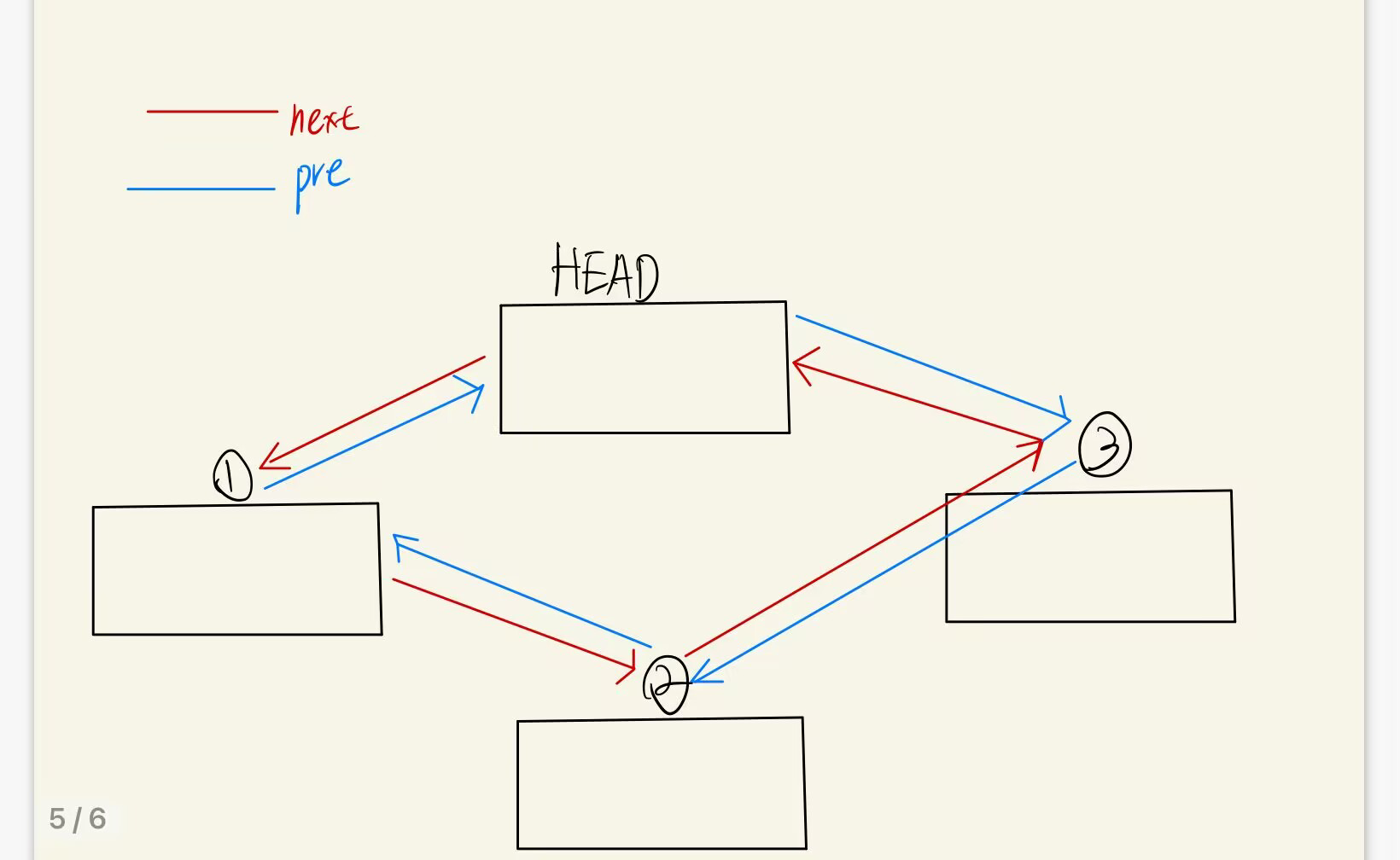
3.1.1 InsertList

函数的声明为bool InsertList(LinkList& L,ElemType data,int pos)，功能是在pos位置后插入data。在本程序中，我们主要使用了pos=0，pos=len两种情况。

这里是pos=0的情况，意味着不断地往head后面插入新的节点，所以最先插入的节点到了最后面。



这里是pos=len的情况，意味着不断地往链表的最后面插入新的节点，所以链表的最后面是最后插入的节点。



3.1.2 QueryList DeleteList

这两个函数查差不多，所以放在一起讲。都是要找到要查询或者删除的位置pos。这里函数的设计有一个特点是会先判断pos是在链表的偏左位置还是偏右边位置。从而选择是用next往下查还是pre往前查。具体的实现以查询为例子

**if(pos <= L.len-pos+1){ //通过后继寻找位置**

**for(int i=1 ;i<=pos ;i++){**

**if(p == NULL) return Error;**

**p = p->next;**

**}**

**}**

**else{ //通过前驱寻找位置**

**for(int i=1 ;i<=L.len-pos+1 ;i++){**

**if(p == NULL) return Error;**

**p = p->prev;**

**}**

**}**

3.2对于大整数的处理

对于输入的大整数，要求先给出所选择的进制，最后都以十进制输出。并且需要给出“，”作为结尾。我们有两种输入方式，一种是直接通过cin从键盘输入，另外一种是通过文件流从文件读入。

**ifstream infile;**

**infile.open("1.txt");**

**infile>>s;**

3.2.1输入

首先我们判断大整数是否未正数，若是，则将head节点的data初始化为1，反之为0。

**if(s[0] == '-'){**

**BigInteger\_Change\_Symbol(L,false);**

**i++;**

**}**

**else BigInteger\_Change\_Symbol(L,true);**

然后我们利用pos=0的插法，将大整数压入双向链表中。

**for(;i<len ;i++){**

**if(s[i] == ','){**

**InsertList(L,data,0);**

**data = 0;**

**}**

**else{**

**data = data\*10 + (s[i]-'0');**

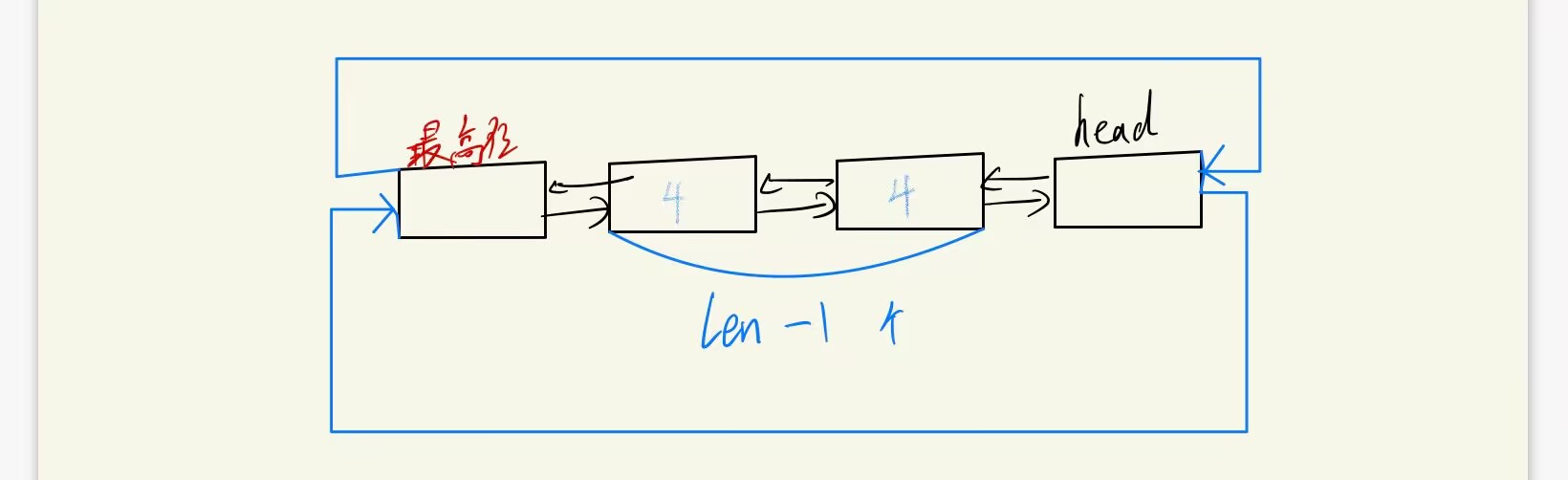
**}**

**}**

**InsertList(L,data,0);**

3.2.2求大整数的位数

这一操作在后面的FFT算法中要用到。由于我们采用万进制储存数据，所以除了最高位以外的每个节点应该是有四位，所以我们计算前面的位数只需要得到（len-1）\*4即可，再找到最高位获得最高位的位数。

****

3.2.3大整数的进位和借位

由于我们采用的是10000进制，所以在采用加法时，当某个节点的数超过了10000，我们需要进行进位。进位的值为当前节点的数据data/10000，然后data更新为data mod 10000。同样的我们也要检查下一位接受了进位的数据后是否超过10000，不断更新。

当我们采用减法时，可能会出现某个节点的数为负数，这时候我们需要向前面借位，借的位数为data/10000+1。同样的，也要检查下一位借完后是否变为负数。

**while(p->next != L.head){**

**if(p->data >= mod){ //需要进位**

**p->next->data += p->data/mod;**

**p->data %= mod;**

**}**

**while(p->data < 0){ //需要借位处理**

**p->next->data--;**

**p->data += mod;**

**}**

**p = p->next;**

**}**

同时我们也要注意，最高位的进位由于要和对下一位进行操作，所以我们把它单独拿出来处理，在链表后面再插入一位。

**if(p->data >= mod){ //最高位进位**

**ElemType add = p->data/mod;**

**p->data %= mod;**

**InsertList(L,add,L.len);**

**}**

由于减法时我们保证是大数减去小数，所以最高位不可能出现负的情况。

3.2.4两数绝对值的比较、

由于大整数的正负只是储存在头节点的data中，所以我们只需要比较各个节点的数据即可。首先比较位数。

if(BigInteger\_Digit(a) != BigInteger\_Digit(b)){

if(BigInteger\_Digit(a) > BigInteger\_Digit(b)) return 1;

else return -1;

}

在位数一样的情况下从head->pre节点开始比较各个节点的大小。

**while(p != a.head && q != b.head){**

**if(p->data > q->data) return 1;**

**if(p->data < q->data) return -1;**

**p = p -> prev;**

**q = q -> prev;**

**}**

**3.3大整数的运算**

**3.3.1加法运算**

**对于加法我们从a,b的正负出发，分为两种情况讨论。**

1. **a<0&&b<0;a>0&&b>0**

**这种情况最简单只需要把c的头节点的data按两种情况设为正负，然后把a b各个节点上的数据相加然后再进位即可。**

1. **a或b有且仅有一个小于0**

**当是b<0时，a+b=a-|b|，这时候我们把b改为正数，然后调用减法就行了。**

**当是a<0时，a+b=b-|a|，把a改成正数然后调用减法，注意调一下a，b在函数参数中的位置。**

**同时我们要注意当a，b对应位置一一相加后，a或b可能还有更高位需要加入c。**

**同时我们要注意的是**

**3.3.2减法运算**

**对于减法我们从a,b的正负出发，分为两种情况讨论。**

**1.a或b有且仅有一个小于0**

**当是a>0 b<0时，a-b=a+|b|，调用加法运算。**

**当是a<0 b>0时，a-b=-（|a|+|b|）,也是调用加法运算。**

**2.a<0&&b<0;a>0&&b>0**

**这时候为了确定a-b的正负，我们就需要知道两者绝对值的大小。**

**当a<0&&b<0时，a-b=-|a|+|b|。**

**当a>0&&b>0时，a-b=|a|-|b|。**

**所以我们只需要比较|a|和|b|的大小就能得到c的正负。同时我们为了程序的方便，将两者中大的赋值给a，小的赋值给b。**

**同时我们也要注意，再一一相减后，大数往往还有多出的位，也要记得把它们添加到c中去。**

**3.3.2 乘法**

**在这里我们计算乘法时，我们可以看作两个多项式相乘。例如对于1234\*321。**

**我们看作**

**参考文献：**

1. **附录：源代码**