**目 录**

[摘 要 3](#_Toc518410444)

[一．排序算法比较问题（2） 4](#_Toc518410445)

[1．采用类语言定义相关的数据类型 4](#_Toc518410446)

[2．算法设计 4](#_Toc518410447)

[3．函数的调用关系图 4](#_Toc518410448)

[4．调试分析 4](#_Toc518410449)

[5．测试结果 4](#_Toc518410450)

[6．源程序（带注释） 4](#_Toc518410451)

[二．递归替换问题（3） 5](#_Toc518410452)

[1．采用类语言定义相关的数据类型 5](#_Toc518410453)

[2．算法设计 5](#_Toc518410454)

[3．函数的调用关系图 5](#_Toc518410455)

[4．调试分析 5](#_Toc518410456)

[5．测试结果 5](#_Toc518410457)

[6．源程序（带注释） 5](#_Toc518410458)

[三．跳马问题（3） 6](#_Toc518410459)

[1．采用类语言定义相关的数据类型 6](#_Toc518410460)

[2．算法设计 6](#_Toc518410461)

[3．函数的调用关系图 6](#_Toc518410462)

[4．调试分析 6](#_Toc518410463)

[5．测试结果 6](#_Toc518410464)

[6．源程序（带注释） 6](#_Toc518410465)

[四．长整数运算问题（4） 7](#_Toc518410466)

[1．采用类语言定义相关的数据类型 7](#_Toc518410467)

[2．算法设计 7](#_Toc518410468)

[3．函数的调用关系图 7](#_Toc518410469)

[4．调试分析 7](#_Toc518410470)

[5．测试结果 7](#_Toc518410471)

[6．源程序（带注释） 7](#_Toc518410472)

[总 结 8](#_Toc518410473)

[参考文献 9](#_Toc518410474)

[致 谢 10](#_Toc518410475)

# 摘 要

本程序主要解决排序算法比较问题，递归替换问题，跳马问题，长整数运算问题。排序算法比较问题是针对实际数据的特点选择合适的排序算法可以使程序获得更高的效率，有时候排序的稳定性还是实际问题中必须考虑的。递归替换问题利用递归算法解决文件替换问题。跳马问题也称为马踏棋盘问题，是算法的设计中的经典问题。在8\*8的方格棋盘中马的行走规则从棋盘的某一方格出发，开始在棋盘上周游，如果能不重复地走遍棋盘上的每一个方格，这样的一条周游路线在数学上被称为国际象棋棋盘上马的哈密尔顿链。长整数运算问题实现两个任意长的正负整数的加减乘除的计算问题。通过该题目的设计过程，可以加深理解线性表的逻辑结构、存储结构，掌握线性表上基本运算的实现。这些程序主要功能是加深我们对算法与数据结构中存储，线性表和栈的理解。让我们对算法与数据结构有个更深刻的认识。

关键词：排序算法比较；递归；跳马；长整数运算；数据结构

文档排版要求：

一级标题：宋体，三号，加粗

二级标题：四号，宋体

正文：宋体，小四号，1．5倍行距

# 一．排序算法比较问题（2）

设计各类排序算法的程序，通过随机的数据测试，比较各算法的关键字比较次数和关键字移动次数。

## 1．采用类语言定义相关的数据类型

typedef struct node {

int data3;

int next;

} node;

int Size; //可排序表的长度

int head;

long compCount; //统计比较次数

long shiftCount; //统计移动次数

## 2．算法设计

a、利用全局变量存储比较和移动次数，用文件存储源数据和比较的数据。

b、大小比较和数值移动使用统一的函数比较和数值移动并对所需要的次数进行叠加，用函数替代所有排序函数中的比较和移动操作。

c、比较之前所有次数清零，返回排序源数据。

## 3．函数的调用关系图

#include”sort”

Initialization()

void sort();

Interpret(int cmd);

结束

图1-3-1 程序调用图

## 4．调试分析

a、程序比较简单，没有太大问题。主要是文件操作时没能找到一个令人心满意足的方案。

b、由于数据量比较大，未作证明测试。

## 5．测试结果

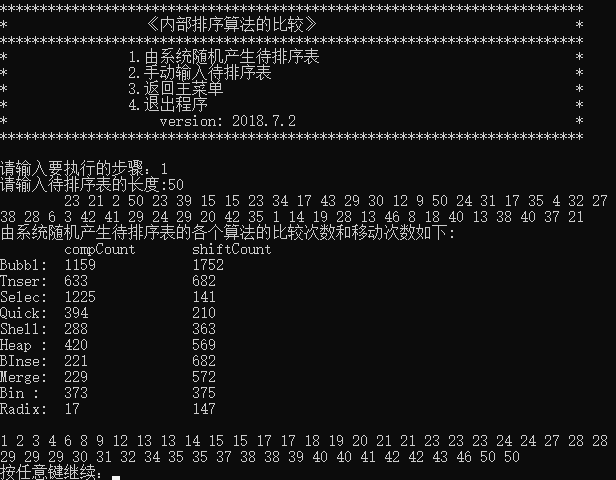


图1-5-1 随机数排序测试结果

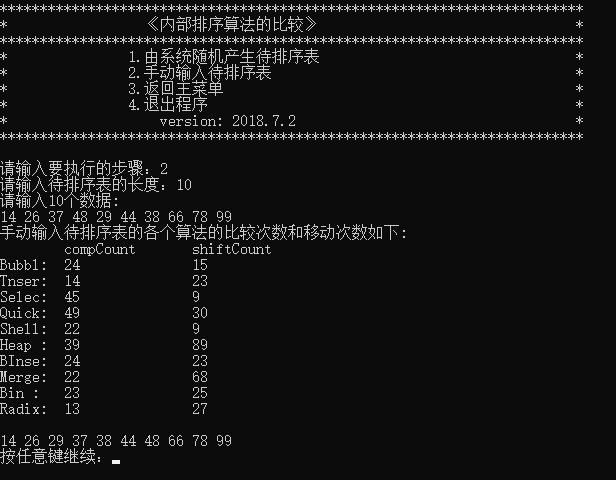


图1-5-2 人工输入排序测试结果

## 6．源程序（带注释）

sort.h

#pragma once

#include<iostream>

#include<ctime>

#include<fstream>

#define MAXSIZE 1000 //可排序表的最大长度

#define SORTNUM 10 //测试10中排序方法

#define max 100 //基数排序时数据的最大位数不超过百位；

typedef int DataType[MAXSIZE + 2];

typedef struct node {

int data3;

int next;

} node;

//对比较次数和移动次数清零

void BeforeSort();

//若表中第i个元素小于第j个元素，则返回True，否则返回False

bool Less(int i, int j);

//交换表中第i个和第j个元素

void Swap(int i, int j);

//将R2[j]赋给R[i]

void Shift(DataType &R, DataType &R2, int i, int j);

void CopyData(DataType list1, DataType list2);

//将可排序表置为逆序

void InverseOrder();

//由系统随机一组数

void RandomizeList();

//输出函数

void output();

//冒泡排序

void BubbleSort();

//插入排序

void InsertSort();

//选择排序

void SelectSort();

int Partition(int low, int high);

//QuickSort的辅助函数

void QSort(int low, int high);

//快速排序

void QuickSort();

//希尔排序

void ShellSort();

//堆排序的调堆函数

void Sift(int left, int right);

//堆排序

void HeapSort();

//折半插入排序

void BInsertSort();

//2-路插入排序

void Binsort();

void Merge(int low, int m, int high);

void MSort(int low, int high);

//归并排序

void MergeSort();

void Distribute(node \*a, int w);

void Collect(node \*a);

//基数排序算法

void RadixSort();

//系统初始化

void Initialization();

//调用各个算法

void Interpret(int cmd);

void sort();

# 三．跳马问题（3）

要求在 64 个国际象棋格子，任意位置放一个马，如何不重复地把格子走完。

## 1．采用类语言定义相关的数据类型

## 2．算法设计

## 3．函数的调用关系图

## 4．调试分析

* 1. 调试中遇到的问题及对问题的解决方法

b、算法的时间复杂度和空间复杂度

## 5．测试结果

## 6．源程序（带注释）

# 四．长整数运算问题（4）

设计程序，实现两个任意长的整数的加、减、乘运算问题。

## 1．采用类语言定义相关的数据类型

//定义长整数类

class LongInt

{

private:

//存储正负

int sign = 1;

//存储数值

std::string number;

};

## 2．算法设计

长整数运算的符号运算法则和整数符号的四则运算相同，所以选用整型保存大数的符号，并进行运算。

字符串是特殊的字符数组，可以存储很长的数据。不考虑内部返回的size()大小，利用C++ 中的auto自动推断int，long等整数存储类型，保证了空间的有效利用。长整数表示可根据生产需求适当修改相关的流操作，此处统一使用连续表示。

所有的相关运算符一律重载，包括加减乘除、大小比较、输入输出流。

加、减、乘运算采用了小学课本的计算法则，为便于运算先对字符串进行了逆置，运算完成后再逆置回。

如果要提高计算效率可进行分组计算，考虑到代码量和时间的问题，该程序不作分组处理。

由于除法未作要求，所以除法使用了减法统计次数的方法获取相关值——商、余数、模，大多电脑每秒减1000次，所以当在1000倍以上时此法不可用于生产环境。

## 3．函数的调用关系图

#include ”LongInt.h”

建立LongInt

对象赋值

根据需求进行加减乘除运算

返回值

结束长整数运算

图4-3-1 函数调用关系图

## 4．调试分析

1. 调试中遇到的问题及对问题的解决方法

b、算法的时间复杂度和空间复杂度

## 5．测试结果

## 6．源程序（带注释）

// LongInt.h

#include<iostream>

#include<string>

//长整数

class LongInt

{

private:

int sign = 1;//正负

std::string number;//数值

bool isbig(std::string& l, std::string& r);

bool compare(LongInt lnum, LongInt rnum);

public:

LongInt(std::string str="0");

auto size() { return number.size(); }

LongInt operator+(LongInt num);

LongInt operator-(LongInt num);

LongInt operator\*(LongInt num);

LongInt operator/(LongInt num);

//LongInt operator%(LongInt num);

const bool& operator==(const LongInt &num);

const bool& operator!=(const LongInt &num);

const bool& operator >(const LongInt &num);

const bool& operator>=(const LongInt &num);

const bool& operator <(const LongInt &num);

const bool& operator<=(const LongInt &num);

//输出长整数

friend std::ostream& operator<<(std::ostream& os, const LongInt &num);

//输入长整数

friend std::istream& operator>>(std::istream& is, LongInt &num);

~LongInt();

};

// LongInt.c

#include "LongInt.h"

//判断左字符串是否大于右字符串

bool LongInt::isbig(std::string& l, std::string& r)

{

if (l.size() == r.size())

return l > r;

return l.size() > r.size() ? true : false;

}

//左值大Ture，相等、小false

bool LongInt::compare(LongInt lnum, LongInt rnum)

{

if (lnum.sign == rnum.sign) {

if (lnum.sign > 0) {

if (isbig(lnum.number, rnum.number))return true;

}

else if (isbig(rnum.number, lnum.number))return true;

}

else if (lnum.sign > rnum.sign)return true;

return false;

}

LongInt::LongInt(std::string str)

{

if (str.front() == '-') {

sign = -1;

str.erase(0, 1);

}

else if (str.front() == '+')str.erase(0, 1);

number = str;

}

LongInt LongInt::operator+(LongInt num)

{

LongInt lint;

std::string l, r, small;

auto lsize = (\*this).size(), rsize = num.size();

auto max = lsize > rsize ? lsize : rsize;

auto temp = 0;

unsigned i = 0;

if (sign == num.sign) {

std::copy(number.crbegin(), number.crend(), std::back\_inserter(l));

std::copy(num.number.crbegin(), num.number.crend(), std::back\_inserter(r));

lint.sign = sign;

while (temp || i < max) {

temp += i < lsize ? l[i] - '0' : 0;

temp += i < rsize ? r[i] - '0' : 0;

lint.number += '0' + temp % 10;

temp /= 10;

i++;

}

l = lint.number.erase(0, 1);

lint.number.clear();

std::copy(l.crbegin(), l.crend(), std::back\_inserter(lint.number));

}

else {

if (num.number == number)return lint;

lint = isbig(number, num.number) ? \*this : num;

small = lint.sign == sign ? num.number : number;

auto min = small.size();

while (max > i) {

if (min < ++i)break;

temp = max >= i ? lint.number[max - i] - '0' : 0;

while (temp < small[min - i] - '0') {

int count = 1;

while (max - i >= count) {

if (lint.number[max - i - count] > '0') {

lint.number[max - i - count]--;

temp += std::pow(10, count);

break;

}

count++;

}

}

temp -= small[min - i] - '0';

for (int count = 0; temp > 0 || count == 0; count++) {

lint.number[max - i - count] = temp % 10 + '0';

temp = temp / 10;

}

}

}

while (lint.number[0] == '0')lint.number.erase(0, 1);

return lint;

}

LongInt LongInt::operator-(LongInt num)

{

num.sign = num.sign > 0 ? -1 : 1;

return \*this + num;

}

LongInt LongInt::operator\*(LongInt num)

{

LongInt lint;

lint.sign = sign \* num.sign;

auto lsize = (\*this).size();

auto rsize = num.size();

int temp = 0, k = 1;

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

for (int j = 0; (j < rsize || temp); j++) {

temp += j < rsize ? (number[lsize - i - 1] - '0')\*(num.number[rsize - j - 1] - '0') : 0;

if (j + i < k) {

temp += lint.number[j + i] - '0';

lint.number[j + i] = '0' + temp % 10;

}

else {

lint.number += '0' + temp % 10;

k++;

}

temp /= 10;

}

}

std::string str=lint.number;

lint.number.clear();

std::copy(str.crbegin(), str.crend(), std::back\_inserter(lint.number));

if(lint.number=="0")lint.sign = 1;

return lint;

}

//num为零报错

LongInt LongInt::operator/(LongInt num)

{

LongInt li;

if (\*this == li)return li;

LongInt max(number), min(num.number), add("1");

while (max >= min)

{

max = max - min;

li = li + add;

}

li.sign = sign \* num.sign;

return li;

}

const bool & LongInt::operator==(const LongInt & num) {

return number == num.number&&sign == num.sign ? true : false;

}

const bool & LongInt::operator!=(const LongInt & num) {

return number == num.number&&sign == num.sign ? false : true;

}

const bool & LongInt::operator>(const LongInt & num) {

return compare(\*this, num);

}

const bool & LongInt::operator>=(const LongInt & num) {

return !compare(num, \*this);

}

const bool & LongInt::operator<(const LongInt & num) {

return compare(num, \*this);

}

const bool & LongInt::operator<=(const LongInt & num) {

return !compare(\*this, num);

}

LongInt::~LongInt()

{

}

std::ostream & operator<<(std::ostream & os, const LongInt & num)

{

if (num.sign < 0)os << '-';

os << num.number;

return os;

}

std::istream & operator>>(std::istream & is, LongInt & num)

{

num.sign = 1;

std::string str;

is >> str;

if (str.front() == '-') {

num.sign = -1;

str.erase(0, 1);

}

else if (str.front() == '+')str.erase(0, 1);

num.number = str;

return is;

}

# 总 结

就编写的程序而言，虽然能达到预期的结果，但在运行时所需的时间比较长，而且总体结构还不够简洁，不太容易去理解。许多问题还需要继续研究，许多技术还需要更多的改进。去图书馆借了不少书，也去网上看了些资料，只是对大概的知识有了点了解，但还是很难着手于写代码，后来就按照老师说的，先搞清楚原理，再考虑如何去实现！后来又去上网查看相关资料，又到图书馆借了很多书看，总算有头绪了。但在调试过程中，还是遇到了很多困难，后来通过了很多同学的帮助才把问题解决了。

通过这次的课程设计，让我更好地掌握了递归思想、链表、拓扑排序以及一维数组等知识，这对我以后的学习生涯以及将来步入社会起到很大的帮助。这次课程设计虽然花了我很多时间和精力，但很值得，因为它对我能力提高起到很大帮助。这次课程设计也提醒我以前知识的匮乏，它给我敲响了警钟，让我意识到自己基础的不扎实。同时在这次课设中，我还学会了如何修改错误。

总的来说，这次课设让我学到了很多很多的东西，让我对未来的工作有了信心，也有了一定的了解。

# 参考文献

[1]严蔚敏，吴伟民．《数据结构（C语言版）》．清华大学出版社．

[2]严蔚敏，吴伟民．《数据结构题集（C语言版）》．清华大学出版社．

[3]《DATA STRUCTURE WITH C++》． William Ford，William Topp ．清华大学出版社（影印版）．

[4]谭浩强．《c语言程序设计》． 清华大学出版社．

# 致 谢

课程设计终于告一段落了，努力过后，颇有收获，很多以前不清楚、不熟悉的内容都在这一周的努力中得到了锻炼，感谢老师给予的大量帮助及指导，感谢同学们的帮助，才让我顺利完成了这次的课程设计！通过他们的帮助，我深刻体会到，做程序设计需要大量的查阅相关的资料，并与他人合作才能编写出一段好的程序。

在此，衷心的感谢张老师的辛勤指导，让我认识这个课题、熟悉这个课题并且最后完成这个课题；感谢同学们的互帮互助，提供那么多经典程序供我参考，并且指出我编程过程中出现的许多问题；感谢每个给过我帮助的人员！ 谢谢你们的支持，谢谢你们！