《数据结构》课程实践报告

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 院、系 | 计算机学院 | | 年级专业 | 计科大类 | 姓名 |  | 学号 |  |
| 实验布置日期 | | 2019-10-10 | | 提交  日期 | 2019-10-17 | | 成绩 |  |

课程实践实验04：线性表应用-长整数运算

## 一、问题描述及要求

[问题描述]

输入：俩个长整型（long）的数字。

要求：

（1）将输入的俩个长整型数字用顺序存储结构的线性表存放；

（2）最终设计并实现两个长整数的加、减、乘运算；

（3）以上数据和主要操作都封装到一个类中。

## 二、概要设计

1. **对实验内容的理解和二次概括。**

**定义两个线性表，每位数字占一位，利用加减进退位关系完成运算，乘法则需要两个循环嵌套相加。**

1. **给出系统的功能列表（可以用图示或清单的形式）**

**加，减，乘**

1. **程序运行的界面设计 （可以用图示等方式，如：首先出现屏幕提示，请用户选择输入配置的方式，1从键盘输入活单元坐标2……3….然后用户）**

**请输入第一个数:12345**

**请输入第二个数:23456**

**相加结果为:35801**

**0 11 12 13 14 1相减结果为:11111**

**相乘结果为:0289564320**

## 三、详细设计

开始

输入数字

倒序放入线性表

开始循环每位运算

否

是否进退位

是

向后一位进/退位

反向输出

终止

## 四、实验结果

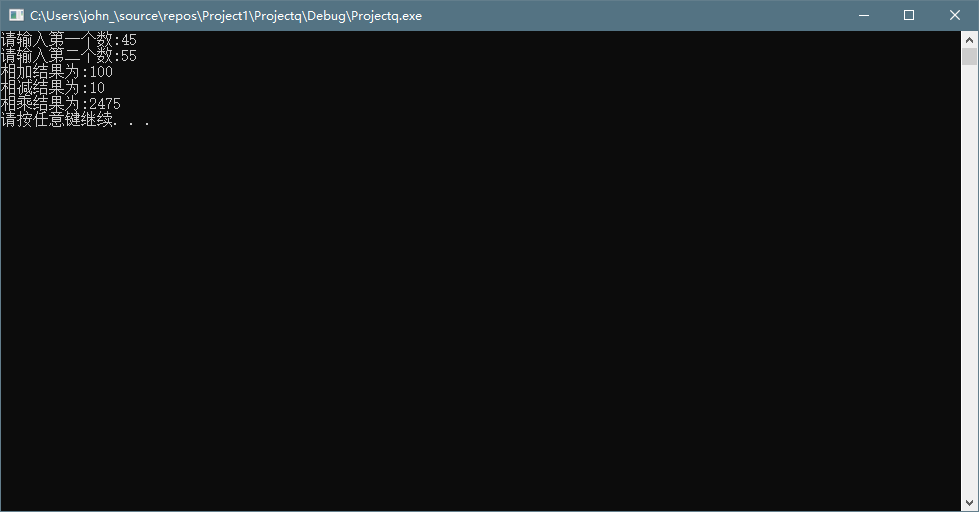
* 测试输入：45 55
* 测试目的：是否能正确进位
* 正确输出：

相加结果为:00

相减结果为:10

相乘结果为:2475

* 实际输出：



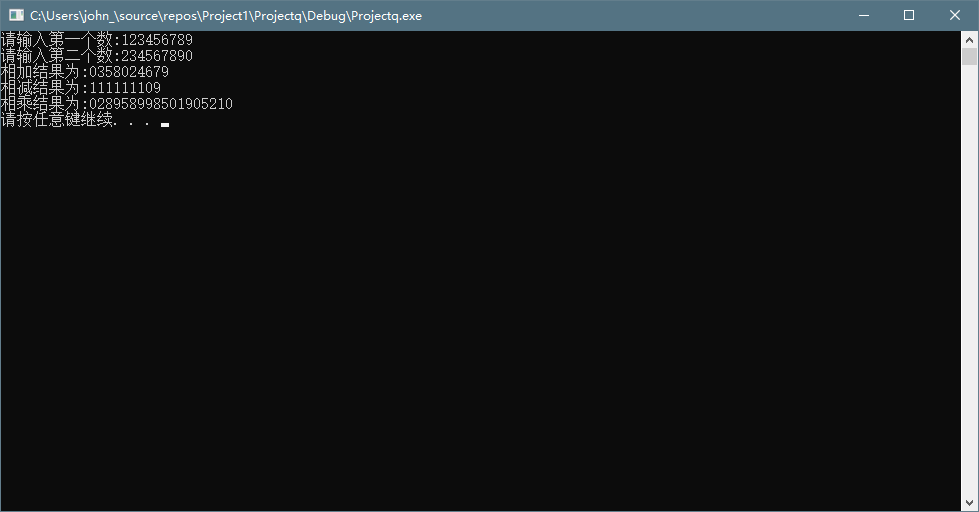
* 测试结论：通过
* 测试输入：123456789 234567890
* 测试目的：是否能正确退位

相加结果为:358024679

相减结果为:111111101

相乘结果为:2895899850190521

* 实际输出：



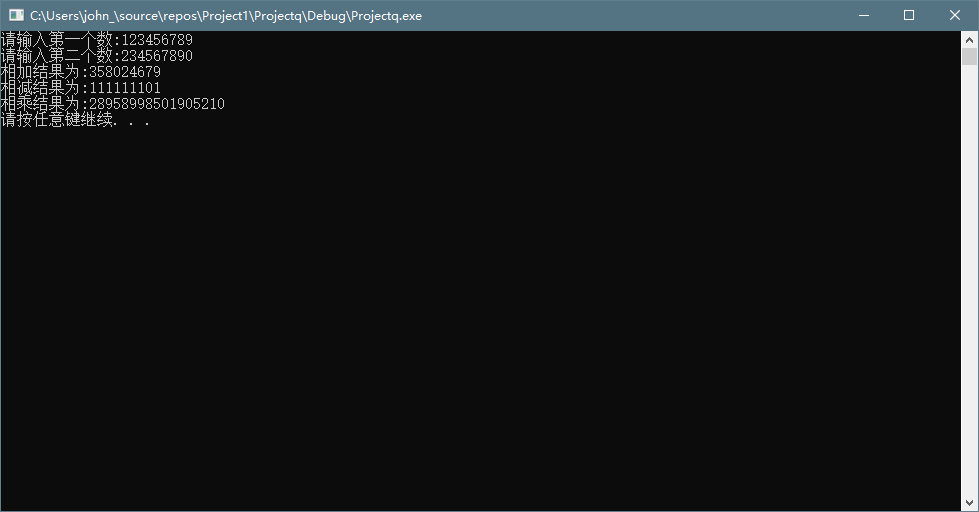
* 测试结论：错误
* 错误原因：退位算法错误，且输出位数多余实际位数，导致数前方多一个0.
* 测试输入：123456789 234567890
* 测试目的：是否能正确退位

相加结果为:358024679

相减结果为:111111101

相乘结果为:28958998501905210

* 实际输出：



* 测试结论：正确

## 五、实验分析与探讨

测试数据的策略：主要测试进退位问题是否解决，做减法时负数必须加10变为正的数并且往前退位，以及输出时输出到哪一位停止，怎样避免出现多余的0.

结果分析：退位问题主要出在没有把负位+10，直接采用了绝对值，这样的做法是错误的。输出时，我发现加法输出位数为max(n1+n2)或max(n1,n2)+1，乘法输出位数为n1+n2-1或n1+n2

算法复杂度：T(n)=O(n2)

## 六、小结

本次实验完成了加减乘法中每位数字的运算，和进退位方法的实践，进退位方法是我碰到的最大难题，需要对此更加熟悉。

## 附录：源代码

#include <iostream>

#include <string>

#include <sstream>

using namespace std;

class calculator

{

int p[201];

int q[201];

int r[201];

int s[201];

int t[402];

string n1, n2;

int n;

public:

calculator(const string &a, const string &b);

~calculator();

void CreateList();

void Plus();

void Minus();

void Multiply();

};

calculator::calculator(const string &a, const string &b)

{

n1 = a;

n2 = b;

memset(p, 0, sizeof(p));

memset(q, 0, sizeof(q));

memset(r, 0, sizeof(r));

memset(s, 0, sizeof(s));

memset(t, 0, sizeof(t));

}

calculator::~calculator()

{

memset(p, 0, sizeof(p));

memset(q, 0, sizeof(q));

memset(r, 0, sizeof(r));

memset(s, 0, sizeof(s));

memset(t, 0, sizeof(t));

}

void calculator::CreateList()

{

stringstream ss;

if (n1.size() > n2.size()) n = n1.size();

else n = n2.size();

for (int i = 0; i < n1.size(); i++)

{

ss << n1[n1.size() - 1 - i];

ss >> p[i];

ss.clear();

}

for (int i = 0; i < n2.size(); i++)

{

ss << n2[n2.size() - 1 - i];

ss >> q[i];

ss.clear();

}

}

void calculator::Plus()

{

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

{

r[i] += (p[i] + q[i]) % 10;

r[i + 1] += (p[i] + q[i]) / 10;

}

for (int i = 0; i < n+1; i++)

{

if (r[i] >= 10)

{

r[i + 1] += r[i] / 10;

r[i] %= 10;

}

}

cout << "相加结果为:";

for (int i = n; i > -1; i--)

{

if (i == n && r[i] == 0) continue;

cout << r[i];

}

cout << endl;

}

void calculator::Minus()

{

int x, y,t,flag=1;

stringstream ss;

ss << n1;

ss >> x;

ss.clear();

ss << n2;

ss >> y;

if (x < y) flag = -1;

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

{

if (flag\*(p[i] - q[i]) < 0) s[i] += 10;

s[i] += flag \* (p[i] - q[i]);

s[i + 1] += (flag\*(p[i] - q[i])-9) / 10;

}

cout << "相减结果为:";

for (int i = n - 1; i > -1; i--)

{

if (i == n - 1 && s[i] == 0) continue;

cout << s[i];

}

cout << endl;

}

void calculator::Multiply()

{

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

{

for (int j = 0; j < n; j++)

{

t[i + j] += (p[i] \* q[j]) % 10;

t[i + j + 1] += (p[i] \* q[j]) / 10;

}

}

for (int i = 0; i < n1.size() + n2.size(); i++)

{

if (t[i] >= 10)

{

t[i + 1] += t[i] / 10;

t[i] %= 10;

}

}

cout << "相乘结果为:";

for (int i = n1.size()+n2.size()-1; i > -1; i--)

{

if (i == n1.size() + n2.size() - 1 && t[i] == 0) continue;

cout << t[i];

}

cout << endl;

}

int main()

{

string s1, s2;

cout << "请输入第一个数:";

cin >> s1;

cout << "请输入第二个数:";

cin >> s2;

calculator c(s1, s2);

c.CreateList();

c.Plus();

c.Minus();

c.Multiply();

system("pause");

}