高精度计算

课程名称 数据结构 成绩评定
实验项目名称 高精度计算 指导老师 干晓聪
实验项目编号 19 实验项目类型 设计性 实验地点 数学系机房
学生姓名 郭彦培 学号 2022101149
学院 信息科学技术学院 系 数学系 专业 信息管理与信息系统
实验时间 2024 年 6 月 13 日上午 ~ 2024 年 7 月 13 日中午

1. 实验目的

利用扩增数组实现对大整数的四则运算

2. 实验环境

计算机：PC X64

操作系统：Windows + Ubuntu20.0LTS

编程语言：C++：GCC std20

IDE：Visual Studio Code

3. 程序原理

模拟手动计算流程即可。

4. 程序代码

4.1. hghCacu.hpp

```cpp
#include <cstdio>
#include <iostream>
#include <cmath>
#include <string>
#include <cstring>
#include <vector>
#include <algorithm>
using namespace std;
class BigInt {
public:
    int sign;
    std::vector<int> v;

    BigInt() : sign(1) {}
    BigInt(const std::string &s) { *this = s; }
    BigInt(int v) {
        char buf[21];
        sprintf(buf, "%d", v);
        *this = buf;
    }
    void zip(int unzip) {
        if (unzip == 0) {
            for (int i = 0; i < (int)v.size(); i++)
                v[i] = get_pos(i * 4) + get_pos(i * 4 + 1) * 10 +
get_pos(i * 4 + 2) * 100 + get_pos(i * 4 + 3) * 1000;
        } else
            for (int i = (v.resize(v.size() * 4), (int)v.size() - 1), a;
i >= 0; i--)
                a = (i % 4 >= 2) ? v[i / 4] / 100 : v[i / 4] % 100, v[i]
= (i & 1) ? a / 10 : a % 10;
        setsign(1, 1);
    }
    int get_pos(unsigned pos) const { return pos >= v.size() ? 0 :
v[pos]; }
    BigInt &setsign(int newsign, int rev) {
        for (int i = (int)v.size() - 1; i > 0 && v[i] == 0; i--)
            v.erase(v.begin() + i);
        sign = (v.size() == 0 || (v.size() == 1 && v[0] == 0)) ? 1 :
(rev ? newsign * sign : newsign);
        return *this;
    }
    std::string to_str() const {
        BigInt b = *this;
        std::string s;
        for (int i = (b.zip(1), 0); i < (int)b.v.size(); ++i)
```

```
            s += char(*(b.v.rbegin() + i) + '0');
        return (sign < 0 ? "-" : "") + (s.empty() ? std::string("0") : s);
    }
    bool absless(const BigInt &b) const {
        if (v.size() != b.v.size()) return v.size() < b.v.size();
        for (int i = (int)v.size() - 1; i >= 0; i--)
            if (v[i] != b.v[i]) return v[i] < b.v[i];
        return false;
    }
    BigInt operator-() const {
        BigInt c = *this;
        c.sign = (v.size() > 1 || v[0]) ? -c.sign : 1;
        return c;
    }
    BigInt &operator=(const std::string &s) {
        if (s[0] == '-')
            *this = s.substr(1);
        else {
            for (int i = (v.clear(), 0); i < (int)s.size(); ++i)
                v.push_back(*(s.rbegin() + i) - '0');
            zip(0);
        }
        return setsign(s[0] == '-' ? -1 : 1, sign = 1);
    }
    bool operator<(const BigInt &b) const {
        return sign != b.sign ? sign < b.sign : (sign == 1 ? absless(b) : b.absless(*this));
    }
    bool operator==(const BigInt &b) const { return v == b.v && sign == b.sign; }
    BigInt &operator+=(const BigInt &b) {
        if (sign != b.sign) return *this = (*this) - -b;
        v.resize(std::max(v.size(), b.v.size()) + 1);
        for (int i = 0, carry = 0; i < (int)b.v.size() || carry; i++) {
            carry += v[i] + b.get_pos(i);
            v[i] = carry % 10000, carry /= 10000;
        }
        return setsign(sign, 0);
    }
    BigInt operator+(const BigInt &b) const {
        BigInt c = *this;
        return c += b;
    }
    void add_mul(const BigInt &b, int mul) {
        v.resize(std::max(v.size(), b.v.size()) + 2);
        for (int i = 0, carry = 0; i < (int)b.v.size() || carry; i++) {
            carry += v[i] + b.get_pos(i) * mul;
            v[i] = carry % 10000, carry /= 10000;
        }
```

```
    }
    BigInt operator-(const BigInt &b) const {
        if (b.v.empty() || b.v.size() == 1 && b.v[0] == 0) return *this;
        if (sign != b.sign) return (*this) + -b;
        if (absless(b)) return -(b - *this);
        BigInt c;
        for (int i = 0, borrow = 0; i < (int)v.size(); i++) {
            borrow += v[i] - b.get_pos(i);
            c.v.push_back(borrow);
            c.v.back() -= 10000 * (borrow >>= 31);
        }
        return c.setsign(sign, 0);
    }
    BigInt operator*(const BigInt &b) const {
        if (b < *this) return b * *this;
        BigInt c, d = b;
        for (int i = 0; i < (int)v.size(); i++, d.v.insert(d.v.begin(),
0))
            c.add_mul(d, v[i]);
        return c.setsign(sign * b.sign, 0);
    }
    BigInt operator/(const BigInt &b) const {
        BigInt c, d;
        BigInt e=b;
        e.sign=1;

        d.v.resize(v.size());
        double db = 1.0 / (b.v.back() + (b.get_pos((unsigned)b.v.size() -
2) / 1e4) +
                           (b.get_pos((unsigned)b.v.size() - 3) + 1) /
1e8);
        for (int i = (int)v.size() - 1; i >= 0; i--) {
            c.v.insert(c.v.begin(), v[i]);
            int m = (int)((c.get_pos((int)e.v.size()) * 10000 +
c.get_pos((int)e.v.size() - 1)) * db);
            c = c - e * m, c.setsign(c.sign, 0), d.v[i] += m;
            while (!(c < e))
                c = c - e, d.v[i] += 1;
        }
        return d.setsign(sign * b.sign, 0);
    }
    BigInt operator%(const BigInt &b) const { return *this - *this / b *
b; }
    bool operator>(const BigInt &b) const { return b < *this; }
    bool operator<=(const BigInt &b) const { return !(b < *this); }
    bool operator>=(const BigInt &b) const { return !(*this < b); }
    bool operator!=(const BigInt &b) const { return !(*this == b); }
};
```

4.2. _PRIV_TEST.cpp

```cpp
#include <Dev\19\hghCacu.hpp>

int main()
{
    string aa,bb;
    cin >> aa >> bb;
    BigInt a = BigInt(aa);
    BigInt b = BigInt(bb);
    cout <<( a + b).to_str() << '\n';
    cout << (a - b ).to_str()<< '\n';
    cout << (a * b ).to_str()<< '\n';
    system("pause");
    return 0;
}
```

5. 测试数据与运行结果

运行上述_PRIV_TEST.cpp 测试代码中的正确性测试模块，得到以下内容：

```
1111111111111111111111111111111111111111
2222222222222222222222222222222222222222
3333333333333333333333333333333333333333
-1111111111111111111111111111111111111111
2469135802469135802469135802469135802468641975308641975308641975308641975308642
```

可以看出，代码运行结果与预期相符，可以认为代码正确性无误。