



**《程序设计原理》实验报告**

**——问题 E: 更大的整数**

**学 号 3020205094**

**姓 名 刘宏伟**

**学 院 智能与计算学部**

**年 级 2021级**

**任课教师 杜朴风**

**2022年 12月 11日**

# 实验目的

掌握C++面向对象编程思想，理解高精度运算的基本实现方法，这里我使用了vector容器及其提供的接口来实现大整数类。

# 二．实验内容

1. **设计思路**

实现MyInteger类的各个构造函数与析构函数，完成getnum成员函数，并重载+-\*这三个运算符，其中考虑使用vector反向存储各位数，便于计算和存取，构造函数部分可复用。

1. **代码实现**（写上必要的注释）

class MyInteger {

private:

// 用vector容器来管理内存空间

vector<int> num;

public:

MyInteger() : MyInteger("0") {}

MyInteger(vector<int> n) {

// 去除前导0

while (n.back() == 0)

n.pop\_back();

if (n.empty())

n.push\_back(0);

num = n;

}

MyInteger(string n) {

// 去除前导0

int nl = n.length(), i = 0;

while (i < nl && n[i] == '0')

i++;

string x = n.substr(i);

if (x.empty())

x = "0";

int xl = x.length();

for (int i = xl - 1; i >= 0; i--) {

num.push\_back(x[i] - '0');

}

}

MyInteger(const char \*n) : MyInteger((string)n) {}

MyInteger(int num) : MyInteger(to\_string(num)) {}

~MyInteger() {}

string getnum() {

string ans = "";

// 由于是逆向存储，输出时使用反向迭代器遍历

for (vector<int>::reverse\_iterator it = num.rbegin(); it != num.rend(); it++) {

ans += to\_string(\*it);

}

return ans;

}

// 由于num为私有变量，提供这个函数来供重载运算符获取容器长度

int getsize() const {

return num.size();

}

// 由于num为私有变量，提供这个函数来供重载运算符获取容器元素

int getpos(int pos) const {

return num.at(pos);

}

// 重载+运算符

MyInteger operator+(const MyInteger &bint2) {

// c为进位，以下为模拟加法运算

int c = 0, l = 0;

vector<int> temp;

while (c || l < num.size() || l < bint2.getsize()) {

if (l < num.size())

c += num.at(l);

if (l < bint2.getsize())

c += bint2.getpos(l);

temp.push\_back(c % 10);

c /= 10;

l++;

}

MyInteger res(temp);

return res;

}

// 重载-运算符

MyInteger operator-(const MyInteger &bint2) {

// b为借位，以下为模拟减法运算，由题意不考虑负数

int b = 0, l = 0;

vector<int> temp;

while (l < num.size()) {

int t = num.at(l) - b + 10;

if (l < bint2.getsize())

t -= bint2.getpos(l);

temp.push\_back(t % 10);

b = 1 - t / 10;

l++;

}

MyInteger res(temp);

return res;

}

// 重载\*运算符

MyInteger operator\*(const MyInteger &bint2) {

// c为进位，以下为模拟乘法运算

int c = 0, l = 1;

vector<int> temp;

temp.push\_back(0);

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

c = 0;

for (int j = 0; j < bint2.getsize(); j++) {

if (i + j >= l) {

temp.push\_back(0);

l++;

}

int p = temp.at(i + j);

p += num.at(i) \* bint2.getpos(j) + c;

c = p / 10;

p %= 10;

temp.at(i + j) = p;

}

temp.push\_back(c);

l++;

}

MyInteger res(temp);

return res;

}

};

1. **结果展示**（Oj平台提交记录）



# 三．实验中遇到的问题和解决办法

没有遇到较大的问题，就是写的过程中需要再次复习资料查阅vector的反向迭代器等的使用，说明不太熟练，需要多久练习，然后就是还要注意const的引用也只能调用同为const的成员函数。