学习总结--数据结构.高精度

定义部分

```
struct bign {
    int len, num[MAXN];

    bign () {
        len = 0;
        memset(num, 0, sizeof(num));
    }
    bign (int number) {*this = number;}
    bign (const char* number) {*this = number;}

    void DelZero ();
    void Put ();

    void operator = (int number);
    void operator = (char* number);

    bool operator <  (const bign& b) const;
    bool operator >  (const bign& b) const { return b < *this; }
    bool operator <= (const bign& b) const { return !(b < *this); }
    bool operator >= (const bign& b) const { return !(*this < b); }
    bool operator != (const bign& b) const { return b < *this || *this < b;}
    bool operator == (const bign& b) const { return !(b != *this); }

    void operator ++ ();
    void operator -- ();
    bign operator + (const int& b);
    bign operator + (const bign& b);
    bign operator - (const int& b);
    bign operator - (const bign& b);
    bign operator * (const int& b);
    bign operator * (const bign& b);
    bign operator / (const int& b);
    //bign operator / (const bign& b);
    int operator % (const int& b);
};
```

去除前导零

```
void bign::DelZero () {
    while (len && num[len-1] == 0)
        len--;

    if (len == 0) {
        num[len++] = 0;
    }
}
```

输出

```
void bign::Put () {
    for (int i = len-1; i >= 0; i--)
        printf("%d", num[i]);
}
```

赋值部分

```
void bign::operator = (char* number) {
    len = strlen (number);
    for (int i = 0; i < len; i++)
        num[i] = number[len-i-1] - '0';

    DelZero ();
}

void bign::operator = (int number) {

    len = 0;
    while (number) {
        num[len++] = number%10;
        number /= 10;
    }

    DelZero ();
}
```

关系函数

```
bool bign::operator < (const bign& b) const {
    if (len != b.len)
        return len < b.len;
    for (int i = len-1; i >= 0; i--)
```

```
        if (num[i] != b.num[i])
            return num[i] < b.num[i];
    return false;
}
```

自加自减

```
void bign::operator ++ () {
    int s = 1;

    for (int i = 0; i < len; i++) {
        s = s + num[i];
        num[i] = s % 10;
        s /= 10;
        if (!s) break;
    }

    while (s) {
        num[len++] = s%10;
        s /= 10;
    }
}

void bign::operator -- () {
    if (num[0] == 0 && len == 1) return;

    int s = -1;
    for (int i = 0; i < len; i++) {
        s = s + num[i];
        num[i] = (s + 10) % 10;
        if (s >= 0) break;
    }
    DelZero ();
}
```

加法

```
bign bign::operator + (const int& b) {
    bign a = b;
    return *this + a;
}

bign bign::operator + (const bign& b) {
    int bignSum = 0;
    bign ans;

    for (int i = 0; i < len || i < b.len; i++) {
        if (i < len) bignSum += num[i];
        if (i < b.len) bignSum += b.num[i];

        ans.num[ans.len++] = bignSum % 10;
        bignSum /= 10;
    }

    while (bignSum) {
        ans.num[ans.len++] = bignSum % 10;
        bignSum /= 10;
    }

    return ans;
}
```

减法

```
bign bign::operator - (const int& b) {
    bign a = b;
    return *this - a;
}

bign bign::operator - (const bign& b) {
    int bignSub = 0;
    bign ans;
    for (int i = 0; i < len || i < b.len; i++) {
        bignSub += num[i];
        bignSub -= b.num[i];
        ans.num[ans.len++] = (bignSub + 10) % 10;
        if (bignSub < 0) bignSub = -1;
    }
    ans.DelZero ();
    return ans;
}
```

乘法

```
bign bign::operator * (const int& b) {
    int bignSum = 0;
    bign ans;
    ans.len = len;
    for (int i = 0; i < len; i++) {
        bignSum += num[i] * b;
        ans.num[i] = bignSum % 10;
        bignSum /= 10;
    }

    while (bignSum) {
        ans.num[ans.len++] = bignSum % 10;
        bignSum /= 10;
    }

    return ans;
}

bign bign::operator * (const bign& b) {
    bign ans;
    ans.len = 0;

    for (int i = 0; i < len; i++){
        int bignSum = 0;

        for (int j = 0; j < b.len; j++){
            bignSum += num[i] * b.num[j] + ans.num[i+j];
            ans.num[i+j] = bignSum % 10;
            bignSum /= 10;
        }
        ans.len = i + b.len;

        while (bignSum){
            ans.num[ans.len++] = bignSum % 10;
            bignSum /= 10;
        }
    }
    return ans;
}
```

除法

```
bign bign::operator / (const int& b) {

    bign ans;

    int s = 0;
    for (int i = len-1; i >= 0; i--) {
        s = s * 10 + num[i];
        ans.num[i] = s/b;
        s %= b;
    }

    ans.len = len;
    ans.DelZero ();
    return ans;
}
```

取模

```
int bign::operator % (const int& b) {

    bign ans;

    int s = 0;
    for (int i = len-1; i >= 0; i--) {
        s = s * 10 + num[i];
        ans.num[i] = s/b;
        s %= b;
    }

    return s;
}
```