OmniInt.h 高精度整数库说明文档

简介

`OmniInt.h` 是一个用于高精度整数计算的C++头文件，它提供了一个 `OmniInt` 类，用于处理超出标准整型范围的任意大小的整数。该库实现了基本的算术运算、比较运算、赋值运算以及类型转换等功能，并支持从字符串和 `long long` 类型进行构造和赋值。此版本为 `OmniInt` 库的 `1.1` 版本。

目录

1. 构造函数

`OmniInt` 类提供了多种构造函数，允许从不同类型的数据创建 `OmniInt` 对象。

OmniInt()

描述: 默认构造函数，创建一个值为0的 `OmniInt` 对象。

示例:

```
OmniInt num1;
// num1 的值为 0
```

OmniInt(long long n)

描述: 从 `long long` 类型构造 `OmniInt` 对象。

参数:

- `n`：`long long` 类型整数，用于初始化 `OmniInt` 对象。

示例:

```
OmniInt num2(123456789012345LL);
// num2 的值为 123456789012345
OmniInt num3(-98765432109876LL);
// num3 的值为 -98765432109876
```

OmniInt(const std::string &s)

描述: 从字符串构造 `OmniInt` 对象。字符串可以包含可选的 `'+'` 或 `'-'` 前缀。

参数:

- `s`：`const std::string &` 类型，表示一个数字字符串。字符串可以以 `'+'` 或 `'-'` 开头，后面跟着数字字符。如果字符串为空或只包含符号，将抛出 `std::invalid_argument` 异常。如果字符串包含非数字字符（除了开头的符号），也将抛出 `std::invalid_argument` 异常。

示例:

```
OmniInt num4("12345678901234567890");
// num4 的值为 12345678901234567890
OmniInt num5("-9876543210987654321");
// num5 的值为 -9876543210987654321
OmniInt num6("+500");
// num6 的值为 500
```

OmniInt(const OmniInt &other)

描述: 拷贝构造函数，从另一个 `OmniInt` 对象构造 `OmniInt` 对象。

参数:

- `other`：`const OmniInt &` 类型，要拷贝的 `OmniInt` 对象。

示例:

```
OmniInt original("1000");
OmniInt copied(original);
// copied 的值为 1000
```

2. 赋值运算符

`OmniInt` 类重载了赋值运算符，允许将 `long long`、`std::string` 或另一个 `OmniInt` 对象赋值给 `OmniInt` 对象。

OmniInt &operator=(long long n)

描述: 将 `long long` 类型整数赋值给 `OmniInt` 对象。

参数:

- `n`：`long long` 类型整数。

返回值: 对当前 `OmniInt` 对象的引用。

示例:

```
OmniInt num;
num = 12345LL;
// num 的值为 12345
```

OmniInt &operator=(const std::string &s)

描述: 将字符串赋值给 `OmniInt` 对象。字符串可以包含可选的 `'+'` 或 `'-'` 前缀。

参数:

- `s`：`const std::string &` 类型，表示一个数字字符串。与构造函数类似，字符串格式不正确会抛出 `std::invalid_argument` 异常。

返回值: 对当前 `OmniInt` 对象的引用。

示例:

```
OmniInt num;
num = "-543210987654321";
// num 的值为 -543210987654321
```

OmniInt &operator=(const OmniInt &other)

描述: 将另一个 `OmniInt` 对象赋值给当前 `OmniInt` 对象。

参数:

- `other`：`const OmniInt &` 类型，用于赋值的 `OmniInt` 对象。

返回值: 对当前 `OmniInt` 对象的引用。

示例:

```
OmniInt a("100");
OmniInt b;
b = a;
// b 的值为 100
```

3. 一元算术运算符

OmniInt operator-() const

描述: 一元负号运算符，返回当前 `OmniInt` 对象的相反数。如果当前对象为0，则返回0。

返回值: 一个新的 `OmniInt` 对象，其值为当前对象的相反数。

示例:

```
 OmniInt num("123");
OmniInt neg_num = -num;
// neg_num 的值为 -123
OmniInt zero(0);
OmniInt neg_zero = -zero;
// neg_zero 的值为 0
```

4. 二元算术运算符

`OmniInt` 类重载了常用的二元算术运算符，支持 `OmniInt` 对象之间的加、减、乘、除和取模运算。

OmniInt operator+(const OmniInt &other) const

描述: 加法运算符，计算两个 `OmniInt` 对象的和。

参数:

- `other`：`const OmniInt &` 类型，加数。

返回值: 一个新的 `OmniInt` 对象，表示两个数的和。

示例:

```
 OmniInt a("100");
OmniInt b("200");
OmniInt sum = a + b;
// sum 的值为 300
OmniInt c("-50");
OmniInt d("30");
OmniInt sum2 = c + d;
// sum2 的值为 -20
```

OmniInt operator-(const OmniInt &other) const

描述: 减法运算符，计算两个 `OmniInt` 对象的差。

参数:

- `other`：`const OmniInt &` 类型，减数。

返回值: 一个新的 `OmniInt` 对象，表示两个数的差。

示例:

```
 OmniInt a("200");
OmniInt b("100");
OmniInt diff = a - b;
// diff 的值为 100
OmniInt c("50");
OmniInt d("80");
OmniInt diff2 = c - d;
// diff2 的值为 -30
```

OmniInt operator*(const OmniInt &other) const

描述: 乘法运算符，计算两个 `OmniInt` 对象的积。

参数:

- `other`：`const OmniInt &` 类型，乘数。

返回值: 一个新的 `OmniInt` 对象，表示两个数的积。

示例:

```
 OmniInt a("10");
OmniInt b("20");
OmniInt prod = a * b;
// prod 的值为 200
OmniInt c("-5");
OmniInt d("10");
OmniInt prod2 = c * d;
// prod2 的值为 -50
```

OmniInt operator/(const OmniInt &other) const

描述: 除法运算符，计算两个 `OmniInt` 对象的商（向零取整）。

参数:

- `other`：`const OmniInt &` 类型，除数。如果除数为0，将抛出 `std::runtime_error` 异常。

返回值: 一个新的 `OmniInt` 对象，其值为两个数相除的商。

示例:

```
 OmniInt a("100");
OmniInt b("3");
OmniInt quot = a / b;
// quot 的值为 33
OmniInt c("-10");
OmniInt d("3");
OmniInt quot2 = c / d;
// quot2 的值为 -3
```

OmniInt operator%(const OmniInt &other) const

描述: 取模运算符，计算两个 `OmniInt` 对象的余数。

参数:

- `other`：`const OmniInt &` 类型，模数。如果模数为0，将抛出 `std::runtime_error` 异常。

返回值: 一个新的 `OmniInt` 对象，其值为两个数相除的余数。

示例:

```
 OmniInt a("100");
OmniInt b("3");
OmniInt rem = a % b;
// rem 的值为 1
OmniInt c("-10");
OmniInt d("3");
OmniInt rem2 = c % d;
// rem2 的值为 -1 (与被除数符号相同)
```

5. 复合赋值运算符

`OmniInt` 类重载了复合赋值运算符，提供便捷的算术运算和赋值的组合操作。

OmniInt &operator+=(const OmniInt &other)

描述: 加法复合赋值运算符，将 `other` 加到当前对象上。

参数:

- `other`：`const OmniInt &` 类型，加数。

返回值: 对当前 `OmniInt` 对象的引用。

示例:

```
 OmniInt num("100");
num += "50";
// num 的值为 150
```

OmniInt &operator-=(const OmniInt &other)

描述: 减法复合赋值运算符，将 `other` 从当前对象中减去。

参数::

- `other`：`const OmniInt &` 类型，减数。

返回值: 对当前 `OmniInt` 对象的引用。

示例:

```
 OmniInt num("100");
num -= "50";
// num 的值为 50
```

OmniInt &operator*=(const OmniInt &other)

描述: 乘法复合赋值运算符，将当前对象乘以 `other`。

参数:

- `other`：`const OmniInt &` 类型，乘数。

返回值: 对当前 `OmniInt` 对象的引用。

示例:

```
 OmniInt num("10");
num *= "5";
// num 的值为 50
```

OmniInt &operator/=(const OmniInt &other)

描述: 除法复合赋值运算符，将当前对象除以 `other`。

参数:

- `other`：`const OmniInt &` 类型，除数。如果除数为0，将抛出 `std::runtime_error` 异常。

返回值: 对当前 `OmniInt` 对象的引用。

示例:

```
OmniInt num("100");
num /= "3";
// num 的值为 33
```

OmniInt &operator%=(const OmniInt &other)

描述: 取模复合赋值运算符，将当前对象对 `other` 取模。

参数:

- `other`：`const OmniInt &` 类型，模数。如果模数为0，将抛出 `std::runtime_error` 异常。

返回值: 对当前 `OmniInt` 对象的引用。

示例:

```
OmniInt num("100");
num %= "3";
// num 的值为 1
```

6. 增量和减量运算符

`OmniInt` 类重载了前缀和后缀的增量（`++`）和减量（`--`）运算符。

OmniInt &operator++()

描述: 前缀自增运算符，将当前 `OmniInt` 对象的值增加1，并返回增加后的引用。

返回值: 对自增后当前 `OmniInt` 对象的引用。

示例:

```
OmniInt num("10");
++num;
// num 的值为 11
```

OmniInt &operator--()

描述: 前缀自减运算符，将当前 `OmniInt` 对象的值减少1，并返回减少后的引用。

返回值: 对自减后当前 `OmniInt` 对象的引用。

示例:

```
OmniInt num("10");
--num;
// num 的值为 9
```

OmniInt operator++(int)

描述: 后缀自增运算符，将当前 `OmniInt` 对象的值增加1，并返回增加前的值。

返回值: 一个新的 `OmniInt` 对象，其值为操作前的值。

示例:

```
OmniInt num("10");
OmniInt old_num = num++;
// num 的值为 11, old_num 的值为 10
```

OmniInt operator--(int)

描述: 后缀自减运算符，将当前 `OmniInt` 对象的值减少1，并返回减少前的值。

返回值: 一个新的 `OmniInt` 对象，其值为操作前的值。

示例:

```
OmniInt num("10");
OmniInt old_num = num--;
// num 的值为 9, old_num 的值为 10
```

7. 关系运算符

`OmniInt` 类重载了所有标准的关系运算符，用于比较两个 `OmniInt` 对象的大小。

bool operator<(const OmniInt &other) const

描述: 小于运算符，检查当前对象是否小于 `other`。

参数:

- `other`：`const OmniInt &` 类型，用于比较的 `OmniInt` 对象。

返回值: 如果当前对象小于 `other`，则返回 `true`，否则返回 `false`。

示例:

```
 OmniInt a("10");
OmniInt b("20");
bool result = (a < b);
// result 为 true
```

bool operator>(const OmniInt &other) const

描述: 大于运算符，检查当前对象是否大于 `other`。

参数:

- `other`：`const OmniInt &` 类型，用于比较的 `OmniInt` 对象。

返回值: 如果当前对象大于 `other`，则返回 `true`，否则返回 `false`。

示例:

```
 OmniInt a("10");
OmniInt b("20");
bool result = (b > a);
// result 为 true
```

bool operator<=(const OmniInt &other) const

描述: 小于等于运算符，检查当前对象是否小于或等于 `other`。

参数:

- `other`：`const OmniInt &` 类型，用于比较的 `OmniInt` 对象。

返回值: 如果当前对象小于或等于 `other`，则返回 `true`，否则返回 `false`。

示例:

```
 OmniInt a("10");
OmniInt b("10");
bool result = (a <= b);
// result 为 true
```

bool operator>=(const OmniInt &other) const

描述: 大于等于运算符，检查当前对象是否大于或等于 `other`。

参数:

- `other`：`const OmniInt &` 类型，用于比较的 `OmniInt` 对象。

返回值: 如果当前对象大于或等于 `other`，则返回 `true`，否则返回 `false`。

示例:

```
 OmniInt a("10");
OmniInt b("10");
bool result = (a >= b);
// result 为 true
```

bool operator==(const OmniInt &other) const

描述: 等于运算符，检查当前对象是否等于 `other`。

参数:

- `other`：`const OmniInt &` 类型，用于比较的 `OmniInt` 对象。

返回值: 如果当前对象等于 `other`，则返回 `true`，否则返回 `false`。

示例:

```
OmniInt a("10");
OmniInt b("10");
bool result = (a == b);
// result 为 true
```

bool operator!=(const OmniInt &other) const

描述: 不等于运算符，检查当前对象是否不等于 `other`。

参数:

- `other`：`const OmniInt &` 类型，用于比较的 `OmniInt` 对象。

返回值: 如果当前对象不等于 `other`，则返回 `true`，否则返回 `false`。

示例:

```
OmniInt a("10");
OmniInt b("20");
bool result = (a != b);
// result 为 true
```

8. 其他成员函数

`OmniInt` 类提供了一些实用函数，用于类型转换和获取数字信息。

long long toLongLong() const

描述: 将 `OmniInt` 对象转换为 `long long` 类型。如果 `OmniInt` 对象的值超出 `long long` 的范围，将抛出 `std::overflow_error` 异常。

返回值: `long long` 类型，表示 `OmniInt` 对象的值。

示例:

```
 OmniInt num("12345");
long long val = num.toLongLong();
// val 的值为 12345

OmniInt large_num("9223372036854775808"); // 超过 long long 最大值
try {
    long long val_overflow = large_num.toLongLong();
} catch (const std::overflow_error& e) {
    std::cerr << e.what() << std::endl; // 输出: OmniInt value too large for
long long
}
```

std::string toString() const

描述: 将 `OmniInt` 对象转换为其字符串表示形式。

返回值: `std::string` 类型，表示 `OmniInt` 对象的值（包含符号，如果为负数）。

示例:

```
 OmniInt num("-9876543210");
std::string str = num.toString();
// str 的值为 "-9876543210"
```

size_t digitCount() const

描述: 获取 `OmniInt` 对象的数字位数（不包括符号）。对于0，返回1。

返回值: `size_t` 类型，表示数字的位数。

示例:

```
 OmniInt num1("12345");
size_t count1 = num1.digitCount();
// count1 的值为 5
OmniInt num2("-987");
size_t count2 = num2.digitCount();
// count2 的值为 3
OmniInt num3(0);
size_t count3 = num3.digitCount();
// count3 的值为 1
```

9. 友元函数 (流运算符)

`OmniInt` 类重载了输入输出流运算符，允许直接使用 `std::cout` 和 `std::cin` 进行 `OmniInt` 对象的输出和输入。

std::ostream &operator<<(std::ostream &os, const OmniInt &n)

描述: 输出流运算符，将 `OmniInt` 对象的值写入输出流。

参数:

- `os`: `std::ostream &` 类型，输出流。
- `n`: `const OmniInt &` 类型，要输出的 `OmniInt` 对象。

返回值: 对输出流的引用。

示例:

```
OmniInt num("12345678901234567890123456789O");
std::cout << "The number is: " << num << std::endl;
// 输出: The number is: 12345678901234567890123456789O
```

std::istream &operator>>(std::istream &is, OmniInt &n)

描述: 输入流运算符，从输入流读取字符串并将其转换为 `OmniInt` 对象。

参数:

- `is`: `std::istream &` 类型，输入流。
- `n`: `OmniInt &` 类型，用于存储读取值的 `OmniInt` 对象。

返回值: 对输入流的引用。

示例:

```
OmniInt num;
std::cout << "Enter a large number: ";
std::cin >> num;
std::cout << "You entered: " << num << std::endl;
// 用户输入 "-99999999999999999999999999999"
// 输出: You entered: -99999999999999999999999999999
```

10. 全局函数 (sqrt)

OmniInt sqrt(const OmniInt &n)

描述: 计算 `OmniInt` 对象的整数平方根。使用牛顿迭代法计算并返回向下取整的整数平方根。

参数:

- `n`：`const OmniInt &` 类型，要计算平方根的非负 `OmniInt` 对象。如果 `n` 为负数，将抛出 `std::domain_error` 异常。

返回值: 一个新的 `OmniInt` 对象，表示 `n` 的整数平方根。

示例:

```
 OmniInt num("100");
OmniInt root = sqrt(num);
// root 的值为 10
OmniInt large_num("100000000000000000000000000000000");
OmniInt large_root = sqrt(large_num);
// large_root 的值为 10000000000000000000000000000000
```

作者: SharkyMew **日期:** 2025年7月7日