作业1 191220154 张涵之

typedef struct { int numerator, denominator; bool sub; } Rational;

void reduce(Rational& x) {

int a = x.numerator, b = x.denominator;

while (b) {

int mod = a % b;

a = b;

b = mod;

}

x.numerator /= a;

x.denominator /= a;

}

Rational create\_rational(int n, int d) {

if (d == 0) {

printf("Denominator should not be 0\n");

exit(1);

}

Rational r;

r.numerator = abs(n);

r.denominator = abs(d);

r.sub = (n \* d < 0);

reduce(r);

return r;

}

Rational add(const Rational a, const Rational b) {

Rational result;

int anum, bnum;

if (a.sub) anum = -a.numerator;

else anum = a.numerator;

if (b.sub) bnum = -b.numerator;

else bnum = b.numerator;

int temp = anum \* b.denominator + bnum \* a.denominator;

result.numerator = abs(temp);

result.denominator = a.denominator \* b.denominator;

result.sub = (temp < 0);

reduce(result);

return result;

}

Rational sub(const Rational a, const Rational b) {

Rational result;

int anum, bnum;

if (a.sub) anum = -a.numerator;

else anum = a.numerator;

if (b.sub) bnum = -b.numerator;

else bnum = b.numerator;

int temp = anum \* b.denominator - bnum \* a.denominator;

result.numerator = abs(temp);

result.denominator = a.denominator \* b.denominator;

result.sub = (temp < 0);

reduce(result);

return result;

}

Rational mul(const Rational a, const Rational b) {

Rational result;

result.numerator = a.numerator \* b.numerator;

result.denominator = a.denominator \* b.denominator;

result.sub = a.sub ^ b.sub;

reduce(result);

return result;

}

Rational div(const Rational a, const Rational b) {

if (b.numerator == 0) {

printf("Divide by 0\n");

exit(1);

}

Rational result;

result.numerator = a.numerator \* b.denominator;

result.denominator = a.denominator \* b.numerator;

result.sub = a.sub ^ b.sub;

reduce(result);

return result;

}

void print\_rational(Rational x) {

if (x.sub) printf("-");

if (x.denominator == 1) printf("%d\n", x.numerator);

else printf("%d/%d\n", x.numerator, x.denominator);

}

功能描述：实现了含正负有理数的抽象类型，可以通过提供两个整型数（分子和分母，其中分母不为零）生成有理数，并进行简单的四则运算和打印。

静态结构定义：struct Rational及其成员int numerator, denominator（分别表示有理数的分子和分母）和bool sub（表示有理数的符号正负）。

动态行为定义：函数create\_rational创建有理数，reduce对有理数进行约分，add、sub、mul、div进行有理数的加减乘除运算、print\_rational打印有理数。

外部接口：函数creat\_rational、add/sub/mul/div和print\_rational是可供外部调用的接口，函数reduce对外不可见，只在以上几个函数中被调用，用户可以默认使用上述接口进行操作得到的有理数都已经经过约分化成最简形式。

在不获得源码的情况下，通过阅读文档知道外部接口函数的参数、返回值类型及其功能，则不需要知道ADT内部具体实现就可以使用这个有理数进行开发。如创建有理数需传入整数分子分母且分母不能为零；加减乘除均为双目运算，参数和返回值都是有理数，调用打印函数可以(-)a/b的形式打印有理数。

int main() {

int a, b, c, d;

int ok = scanf("%d%d%d%d", &a, &b, &c, &d);

Rational e = create\_rational(a, b);

Rational f = create\_rational(c, d);

print\_rational(add(e, f));

print\_rational(sub(e, f));

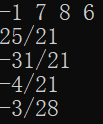
print\_rational(mul(e, f));

print\_rational(div(e, f));

return 0;

}

这段代码让用户输入abcd四个整数，调用create\_rational构造e = a/b，f = c/d这两个有理数，用add/sub/mul/div计算e+f、e-f、e\*f和e/f并打印结果，如图：



代码来源：<https://blog.csdn.net/weixin_43912687/article/details/110591770>，源码对负数情况有欠考虑，且没有提供有理数的格式化打印，我对此稍微做了修改。