
KURS JEZYKA C++

LICZBY WYMIERNE

Instytut Informatyki Uniwersytetu Wrocławskiego

Paweł Rzechonek

Prolog.

Liczba wymierna to taka liczba, którą można zapisać w postaci ułamka zwykłego, czyli w postaci $\frac{p}{q}$, gdzie p to dowolna liczba całkowita a q to liczba całkowita różna od 0. Zbiór wszystkich liczb wymiernych oznaczamy symbolem \mathbb{Q} i formalnie można go zdefiniować jako:

$$\mathbb{Q} = \left\{ \frac{p}{q} : p, q \in \mathbb{Z} \wedge q \neq 0 \right\}$$

Liczby wymierne z operacją dodawania i mnożenia wraz z zerem (element neutransy dodawania) i z jedynką (element neutransy mnożenia) stanowią ciało. Szczególnym przypadkiem liczb wymiernych są liczby całkowite.

Zadanie.

Zdefiniuj klasę *wymierna*, reprezentującą liczbę wymierną w postaci pary liczb całkowitych: licznika i mianownika.

```
class wymierna
{
    int licz, mian;
    // ...
};
```

Zadbaj o to, aby mianownik zawsze był liczbą dodatnią oraz aby największy wspólny dzielnik licznika i mianownika zawsze był równy 1. Udostępnij też gettery, czyli funkcje składowe umożliwiające odczyt licznika i mianownika.

Klasa *wymierna* powinna być wyposażona w konstruktor z licznikiem i mianownikiem, konstruktor konwertujący z wartości typu `int` (możesz zaadoptować do tego celu poprzedni konstruktor definiując drugi argument jako domyślny). Klasa ta ma implementować semantykę kopiowania — musisz rozstrzygnąć, czy konstruktor kopiujący i przypisanie kopiujące zdefiniować samodzielnie czy zdać się na kompilator. Zastanów się też, czy w przypadku klasy *wymierna* będzie potrzebna implementacja semantyki przenoszenia.

W klasie *wymierna* zdefiniuj operatory umożliwiające wykonywanie obliczeń arytmetycznych (dodawanie, odejmowanie, mnożenie i dzielenie), operator `-` do zmiany znaku na przeciwny i operator `!` do wyznaczenia odwrotności (zamiana licznika z mianownikiem z pozostawieniem

znaku liczby w liczniku). Zdefiniuj także operator rzutowania na typ `double` oraz operator jawnego rzutowania na typ `int`.

Nie zapomnij przy każdej funkcji składowej, przy konstruktorach i przy operatorach zadeklarować listy zgłaszanych wyjątków (na przykład, przy dzieleniu przez 0 należy zgłosić wyjątek `dzielenie_przez_0`, a w przypadku operacji arytmetycznych gdy wynik nie będzie mógł być wyrażony dokładnie jako iloraz dwóch liczb typu `int` należy zgłosić wyjątek `przekroczenie_zakresu`). Zaprojektuj całą hierarchię klas wyjątków na potrzeby klasy reprezentującej liczby wymierne zaczynając od klasy bazowej `wyjatek_wymierny`.

Zaprogramuj także zaprzyjaźniony operator zapisania liczby wymiernej do strumienia wyjściowego `operator<<` w postaci liczby rzeczywistej z ułamkiem okresowym.

```
class wymierna
{
    // ...
    friend ostream& operator<< (ostream &wyj, const wymierna &w);
};
```

Na koniec napisz program, który rzetelnie przetestuje wszystkie metody z klasy `wymierna` (wraz ze zgłaszanymi przez nie wyjątkami).

Uzupełnienie.

Definicję klasy `wymierna` umieść w przestrzeni nazw `obliczenia`.

Elementy w programie, na które należy zwracać uwagę.

- Podział programu na pliki nagłówkowe i pliki źródłowe (wyodrębniony osobny plik z funkcją `main()`).
- Umieszczenie definicji klasy `wymierna` w przestrzeni nazw `obliczenia`.
- Definicja operatorów arytmetycznych i konwertujących dla liczb wymiernych.
- Tekstowa reprezentacja liczby wymiernej w postaci ułamka okresowego.
- Definicja hierarchii klas wyjątków.