laboratorium: zadanie 9 *termin: 16–19 maja 2011 r.*

KURS JĘZYKA C++

LICZBY WYMIERNE

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie.

Zdefiniuj klasę `LiczbaWym` reprezentującą liczbę wymierną W klasie tej umieść dwa niepubliczne pola `licznik` i `mianownik` — pola te niech będą liczbami całkowitymi o nieograniczonej precyzji `LiczbaCalk` (skorzystaj z liczb całkowitych z poprzedniego zadania). Zadbaj o to, by w klasie tej mianownik zawsze był dodatni oraz by największy wspólny dzielnik licznika i mianownika był równy 1.

```
class LiczbaWym
{
protected:
    LiczbaCalk liczn, mian;
public:
    LiczbaWym () throw ();
    LiczbaWym (int liczn, int mian=1) throw (badarg);
    LiczbaWym (const LiczbaWym &x) throw ();
    virtual ~LiczbaWym () throw ();
    LiczbaWym & operator= (const LiczbaWym &x) throw ();
public:
    inline LiczbaCalk licznik () const;
    inline LiczbaCalk mianownik () const;
    LiczbaCalk cz_calk () const;
    LiczbaWym cz_ulamk () const;
// ...
};
```

Klasa `LiczbaWym` powinna być wyposażona w konstruktor domyślny (wartość domyślna nowoutworzonej liczby ma wynosić 0), konstruktor inicjalizowany wartościami typu `int`, konstruktor kopiujący, operator przypisania kopiującego i destruktor. Powinieneś też zdefiniować gettery do podawania licznika, mianownika, części całkowitej i części ułamnowej liczby wymiernej. W klasie `LiczbaWym` zdefiniuj operatory umożliwiające wykonywanie obliczeń arytmetycznych (dodawanie, odejmowanie, mnożenie, dzielenie i zmiana znaku).

```
class LiczbaWym
{
// ...
public:
    LiczbaWym operator- () throw ();
    friend LiczbaWym operator+ (const LiczbaWym &x, const LiczbaWym &y) throw ();
    LiczbaWym & operator+= (const LiczbaWym &y) throw ();
    friend LiczbaWym operator/ (const LiczbaWym &x, const LiczbaWym &y) throw (div0);
    LiczbaWym & operator/= (const LiczbaWym &y) throw (div0);
// ...
};
```

Przy konstruowaniu obiektu `LiczbaWym` za pomocą liczb całkowitych zgłoś wyjątek `badarg`, gdy mianownik będzie liczbą ujemną lub zerem. Przy dzieleniu zwróć uwagę na wartość dzielnika i w przypadku próby dzielenia przez 0 zgłoś wyjątek `div0`. Klasy wyjątków `badarg` i `div0` zdefiniuj samodzielnie dziedzicząc po `runtime_error` z biblioteki standardowej.

Zaprogramuj także zaprzyjaźnione operatory do czytania ze stumienia wejściowego `operator>>` i pisania do strumienia wyjściowego `operator<<` liczb wymiernych w postaci dziesiętnej stałopozycyjnej (ułamki okresowe).

```
class LiczbaWym
{
// ...
public:
    friend istream operator>> (istream &we, LiczbaWym &x);
    friend ostream operator<< (ostream &wy, const LiczbaWym &x);
};
```

Umieść definicje klas `LiczbaCalk` i `LiczbaWym` w przestrzeni nazw `obliczenia`. Potem stwórz bibliotekę statyczną albo współdzieloną z obydwiema klasami i nazwij ją `liczby.lib` pod Windowsem albo `libliczby.a` pod Linuxem.

Na koniec napisz program, który korzystając z Twojej biblioteki `liczby` rzetelnie przetestuje wszystkie metody w klasie `LiczbaWym`.

Wskazówka.

Pomocne informacje o ułamkach okresowych znajdziesz na stronach:

- `http://www.mif.pg.gda.pl/kmd/materialy/teoria_liczb/Ulamki%20Okresowe.pdf`
- `http://www.gumienny.edu.pl/materialy-dodatki/jakubas/kl1/4/4.htm`