laboratorium: zadanie 8 *termin: 5–11 maja 2011 r.*

KURS JĘZYKA C++

DŁUGIE LICZBY

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie.

Zdefiniuj klasę `LiczbaCalk` reprezentującą liczbę całkowitą o nieograniczonym zakresie. Liczby te powinny być kodowane binarnie w systemie *uzupełnienia dwójkowego U2*. Do pamiętania długich liczb całkowitych wykorzystaj tablicę bitów z wcześniejszego zadania.

```
class LiczbaCalk
{
protected:
    TabBit *bity;
public:
    LiczbaCalk () throw ();
    LiczbaCalk (int n) throw ();
    LiczbaCalk (const LiczbaCalk &x) throw ();
    virtual ~LiczbaCalk () throw ();
    LiczbaCalk & operator= (const LiczbaCalk &x) throw ();
// ...
};
```

Klasa `LiczbaCalk` powinna być wyposażona w konstruktor domyślny (wartość domyślna nowoutworzonej liczby ma wynosić 0), konstruktor inicjalizowany wartością typu `int`, konstruktor kopiujący, operator przypisania kopiującego i destruktor. Nie zapomnij przy każdej metodzie dadeklarować jakie wyjątki one zgłaszają.

W klasie `LiczbaCalk` zdefiniuj operatory umożliwiające wykonywanie obliczeń arytmetycznych (dodawanie, odejmowanie, mnożenie, dzielenie, reszta z dzielenie i zmiana znaku). Pamiętaj, że klasa reprezentująca długą liczbę może być dość duża i nie powinno się (o ile to nie jest konieczne) przekazywać jej przez wartość za pomocą stosu. Można więc zdefiniować po dwie wersje każdego operatora arytmetycznego: jeden jako funkcja zaprzyjaźniona a drugi jako składowy operator przypisania. Do obliczeń arytmetycznych mogą ci się przydać operatory bitowe oraz przesunięcia.

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

W trakcie obliczeń obliczeń arytmetycznych (w szczególności przy mnożeniu) wartość liczby może szybko rosnąć — powinieneś o tym pamiętać, aby używać odpowiednio dużej tablicy bitów do przechowywania liczby, a w razie konieczności tablice taką należy powiększyć (albo pomniejszyć). Przy dzieleniu zwróć uwagę na wartość dzielnika i w przypadku próby dzielenia przez 0 zgłoś wyjątek `div0`. Klasę wyjątku `div0` zdefiniuj samodzielnie dziedzicząc po `exception` z biblioteki standardowej.

Zaprogramuj także zaprzyjaźnione operatory do czytania ze stumienia wejściowego `operator>>` i pisania do strumienia wyjściowego `operator<<` długich liczb całkowitych w postaci dziesiętnej.

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

Na koniec napisz program, który rzetelnie przetestuje wszystkie metody w klasie `LiczbaCalk`. Program testowy uzupełnij o kod, który obliczy i wypisze wartość 128–mej liczby Fibonacciego i 256–tej liczby Catalana.