laboratorium: zadanie 7 termin: 21–23 kwietnia 2015 r.

KURS JĘZYKA C++

DŁUGIE LICZBY CAŁKOWITE

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 boolowską albo tablicę bitów z wcześniejszego zadania.

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

Klasa `LiczbaCalk` powinna być wyposażona w konstruktor domyślny (wartość domyślna nowo utworzonej liczby ma wynosić 0), konstruktor inicjalizowany wartością typu `int`, konstruktor kopiujący, konstruktor przenoszący, przypisanie kopiujące, przypisanie przenoszące i destruktor. Nie zapomnij przy każdej metodzie zadeklarować wyjątki zgłaszają te metody.

W klasie `LiczbaCalk` zdefiniuj operatory umożliwiające wykonywanie obliczeń arytmetycznych (dodawanie, odejmowanie, mnożenie, dzielenie, reszta z dzielenie, zmiana znaku, inkrementacja i dektementacja). 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:
    friend LiczbaCalk operator+ (const LiczbaCalk &x, const LiczbaCalk &y) noexcept;
    LiczbaCalk& operator+= (const LiczbaCalk &y) noexcept;
    // ...
    friend LiczbaCalk operator/ (const LiczbaCalk &x, const LiczbaCalk &y) throw (div0);
    LiczbaCalk& operator/= (const LiczbaCalk &y) throw (div0);
    LiczbaCalk operator- () const noexcept;
    LiczbaCalk& operator-- () noexcept;
    LiczbaCalk& operator-- (int) noexcept;
    LiczbaCalk& operator++ () noexcept;
    LiczbaCalk& operator++ (int) noexcept;
    // ...
};
```

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 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);
};
```

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

Na koniec napisz program, który rzetelnie przetestuje wszystkie metody w klasie `LiczbaCalk`. Uzupełnieniem programu testowego niech będzie wyliczenie (za pomocą funkcji rekurencyjnej) i wypisanie liczby $100!$.

Uwaga.

Podziel program na pliki nagłówkowe i źródłowe oraz nie używaj w swoim kodzie globalnej dyrektywy `using namespace`.