Laboratorium: zadanie nr 6 *3/6 kwietnia 2006*

JĘZYK PROGRAMOWANIA C++

TABLICA BITÓW

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zdefiniuj klasę `TabBit` reprezentującą tablicę bitów. Najprościej implementuje się taką strukturę danych za pomocą zwykłej tablicy typu `int[]`, przeznaczając na zapamiętanie bitu całe słowo. Jest to rozwiązanie proste, ale bardzo rozrzutne co do zużywanej pamięci — tablica bitów pamiętana w ten sposób jest kilka/kilanaście razy obszerniejsza niż potrzeba. A więc takie rozwiązanie nas nie satysfakcjonuje, szczególnie gdy trzeba posługiwać się w programie wieloma dużymi tablicami (chodzi o tablice zawierające tysiące a nawet miliony bitów).

Należy zatem tak zaprojektować tablice bitowe, aby przydzielona pamięć była wykorzystywana co do bitu (modulo rozmiar słowa). W klasie `TabBit` zdefiniuj operator indeksowania, który umożliwiałby zarówno czytanie z tablicy, jak również pisanie do niej. Oto fragment kodu, który powinien się skompilować i uruchomić:

```
TabBit t(72);          // tablica 72 bitow
t[0] = 1;              // ustawienie bitu 0-ego bitu na 1
t[71] = true;          // ustawienie bitu 71-go bitu na 1
bool b = t[0];         // odczytanie bitu 0-ego
t[40] = b;             // ustawienie bitu 40-go
t[36] = t[38] = t[71]; // przepisanie bitu 71-go do bitow 38-go i 36-go
cout<<t<<endl;         // wysietlenie zawartosci tablicy bitow na ekranie
```

Ponieważ nie można zaadresować pojedynczego bitu (a tym samym nie można ustamowić referencji do niego), więc trzeba się posłużyć specjalną techniką umożliwiającą dostęp do pojedynczego bitu w tablicy. Robi się to poprzez zastosowanie obiektów niewidocznej dla programisty klasy pomocniczej, umiejącej odczytać i zapisać pojedynczy bit.

```
class TabBit
{
    typedef unsigned long long slowo; // komorka w tablicy
    static int rozmiarSlowa; // rozmiar slowa w bitach
    friend istream & operator >> (istream &we, TabBit &tb);
    friend ostream & operator << (ostream &wy, const TabBit &tb);
    class Ref; // klasa pomocnicza dla operatora indeksowania
    class Zakres {}; // klasa wyjatku
protected:
    slowo *tab;
    int dl;
```

```
private:
    void _TabBit (const TabBit &tb);
public:
    explicit TabBit (int rozm);
    TabBit (const TabBit &tb);
    ~TabBit ();
public:
    TabBit & operator = (const TabBit &tb);
private:
    bool czytaj (int i) const;
    bool pisz (int i, bool b);
public:
    bool operator[] (int i) const; // operator indeksowania dla czytania
    Ref operator[] (int i); // operator indeksowania dla pisania
    int rozmiar () const { return dl; } // rozmiar tablicy
};
```

Klasa `Ref` jest klasą pomocniczą, która umożliwia zaimplementowanie operatora indeksowania rozróżniającego czytanie i pisanie w klasie `TabBit`— zastanów się jak powinna ona być zaimplementowana.

Do kompletu podefiniuj operatory koniunkcji, alternatywy i alternatywy wykluczającej (operatory zaprzyjaźnione) operujące na tablicach bitów, oraz operatory przypisania połączone z wymienionymi operatorami bitowymi. Nie zapomnij też o operatorach czytania ze strumienia wejściowego i pisania do strumienia wyjściowego.

Całą definicję klasy `TabBit` podziel na część nagłówkową i źródłową. Następnie w osobnym pliku umieść program testowy, który sprawdzi poprawność zdefiniowanych przez Ciebie operacji na tablicach bitowych i operacji na poszczególnych bitach tych tablic.

Uwaga! Klasa `TabBit` będzie wykorzystana w następnym zadaniu, a zatem warto napisać ją bardzo starannie.