laboratorium: zadanie 4 termin: 24–26 marca 2015 r.

KURS JĘZYKA C++

TABLICA BITÓW

Instytut Informatyki Uniwersytetu Wrocławskiego *Paweł Rzechonek*

Zadanie.

Zdefiniuj klasę `tab_bit` 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 `tab_bit` zdefiniuj operator indeksowania, który umożliwiałby zarówno czytanie z tablicy, jak również pisanie do niej. Oto fragment kodu, który powinien się skompilować i uruchomić:

```
tab_bit t(46);          // tablica 46-bitowa
tab_bit u(45ull);       // tablica 64-bitowa (sizeof(long long)*8)
tab_bit v(t);           // tablica 46-bitowa (skopiowana z t)
tab_bit w(tab_bit(8)); // tablica 8-bitowa (wykradnięta obiektowi tymczasowemu)
v[0] = 1;               // ustawienie bitu 0-go bitu na 1
t[45] = true;           // ustawienie bitu 45-go bitu na 1
bool b = v[1];          // odczytanie bitu 1-go
u[45] = u[46] = u[63]; // przepisanie bitu 63-go do bitow 45-go i 46-go
cout<<t<<endl;          // wysietlenie zawartości tablicy bitów 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 tab_bit
{
    typedef unsigned long long slowo; // komorka w tablicy
    static const int rozmiarSlowa; // rozmiar slowa w bitach
    friend istream & operator >> (istream &we, tab_bit &tb);
    friend ostream & operator << (ostream &wy, const tab_bit &tb);
    class ref; // klasa pomocnicza dla operatora indeksowania
```

```
protected:
    int dl; // liczba bitów
    slowo *tab; // tablica bitów
public:
    explicit tab_bit (int rozm); // wyzerowana tablica bitow [0...rozm]
    explicit tab_bit (slowo tb); // tablica bitów [0...rozmiarSlowa]
                                 // zainicjalizowana wzorcem
    tab_bit (const tab_bit &tb); // konstruktor kopiujący
    tab_bit (tab_bit &&tb); // konstruktor przenoszący
    tab_bit & operator = (const tab_bit &tb); // przypisanie kopiujące
    tab_bit & operator = (tab_bit &&tb); // przypisanie przenoszące
    ~tab_bit (); // destruktor
private:
    bool czytaj (int i) const; // metoda pomocnicza do odczytu bitu
    bool pisz (int i, bool b); // metoda pomocnicza do zapisu bitu
public:
    bool operator[] (int i) const; // indeksowanie dla stałych tablic bitowych
    ref operator[] (int i); // indeksowanie dla zwykłych tablic bitowych
    inline int rozmiar () const; // rozmiar tablicy w bitach
 public:
    // operatory bitowe: | i |=, & i &=, ^ i ^= oraz !
 public:
    // zaprzyjaźnione operatory strumieniowe: << i >>
};
```

Klasa `ref` jest klasą pomocniczą, której zadaniem jest zaadresowanie pojedynczego bitu w tablicy — zastanów się jak powinna ona być zaimplementowana.

Do kompletu podefiniuj operatory koniunkcji, alternatywy, różnicy symetrycznej w połączeniu z przypisaniem oraz operator negacji, które będą wykonywać działania na całych tablicach bitów. Nie zapomnij też o operatorach czytania ze strumienia wejściowego i pisania do strumienia wyjściowego. Niektóre operatory powinny się przyjaźnić z klasą `tab_bit` a pozostałe powinny być jej składowymi.

Na koniec napisz program, który rzetelnie przetestuje klasę `tab_bit` (operacje na poszczególnych bitach tablicy oraz na całych tablicach bitów).

Uwaga.

Podziel program na pliki nagłówkowe i źródłowe. Definicję klasy dla tablicy bitów umieść w pliku `tabbit.hpp` a definicje jej funkcji i operatorów składowych w pliku `tabbit.cpp`. Program główny w pliku `main.cpp` ma rzetelnie przetestować poprawność zdefiniowanych przez Ciebie operacji na tablicach bitowych i operacji na poszczególnych bitach tablicy.

Podpowiedź.

W funkcjach składowych i w konstruktorach zgłaszaj błędy za pomocą instrukcji `throw`.