Operacje na bitach (Kółko Informatyczne, 11 października 2011)

Dzisiaj zapoznamy się z operacjami na bitach. Potrzebne jest do tego rozumienie zapisu binarnego.

**Operatory**

|  |  |  |
| --- | --- | --- |
| Operator | Działanie na bitach | Działanie na bajtach |
| |  “or”  Suma logiczna | 0 | 0 == 0  0 | 1 == 1  1 | 0 == 1  1 | 1 == 1 | 0b1100 | 0b1010 == 0b1110  12 | 10 == 14  0x0C | 0x0A == 0x0E |
| &  “and”  Iloczyn logiczny | 0 & 0 == 0  0 & 1 == 0  1 & 0 == 0  1 & 1 == 1 | 0b1100 | 0b1010 == 0b1000  12 & 10 == 8  0x0C & 0x0A == 0x08 |
| ^  “xor” (exclusive or)  Różnica symetryczna | 0 ^ 0 == 0  0 ^ 1 == 1  1 ^ 0 == 1  1 ^ 1 == 0 | 0b1100 | 0b1010 == 0b0110  12 ^ 10 == 6  0x0C ^ 0x0A == 0x06 |
| >>  Przesunięcie w prawo |  | 0b1000 >> 1 == 0b100  0b1000 >> 2 == 0b10  0b1000 >> 3 == 0b1  0b1000 >> 4 == 0 |
| <<  Przesunięcie w lewo |  | 0b1000 << 1 == 0b10000  0b1000 << 2 == 0b100000 |
| ~  Negacja | ~ 0 == 1  ~ 1 == 0 | ~ 0b11001010 == 0b00110101 |

**Ćwiczenia**

Oblicz wartości wyrażeń:

1. ~0b101110 = 0b…….……. = …………… (dziesiętnie) = 0x……………….
2. 0b101110 & 0b00001111 = …………………..
3. 0b101110 | 0b00001111 = ……………………
4. 53 >> 1 = ……………………..  
   53 >> 2 = ……………………..  
   53 << 1 = ……………………..  
   53 << 3 = ……………………..  
   53 >> 6 = ……………………..
5. Niech n będzie typu char. Kiedy n >> 7 jest równe 1? Kiedy jest równe 0? Czy może przyjąć inne wartości?
6. Zastanów się, co robi z bitami w bajcie odjęcie od niego jedynki. Możesz pomyśleć na przykładach:  
   0b1011000 – 1 = ……………….……, 0b0b101100 – 1 = ……………………..  
   Czy w układzie binarnym działa odejmowanie pisemne? Jak?
7. Niech a = 0b00111100. Oblicz a | (a >> 1) = 0b……………………….  
    b = 0b01000010. Oblicz b | (b >> 1) = 0b……………………….  
    c = 0b01000000. Oblicz c | (c >> 1) = 0b……………………….  
    d = 0b00111100. Oblicz d | (d >> 1) = 0b……………………….  
    e = 0b00000010. Oblicz e | (e >> 1) = 0b……………………….  
    f = 0b01000010. Oblicz f | (f >> 1) = 0b……………………….  
    g = 0b00111100. Oblicz g | (g >> 1) = 0b……………………….

**Tablice**

Tablica to ciąg zmiennych danego typu, do którego elementów można odwołać się podając ich numery. Deklaracja tablicy znaków (bajtów):

char t[10]

Tablica „t” ma dziesięć elementów: t[0], t[1], t[2], t[3], …, t[9]. Ostatni element ma numer „9”, bo zaczynamy numerację od zera.

Do zapoznania się z działaniem tablic wykorzystamy następujący szablon:

#include<fstream>

using namespace std;

int main()

{

fstream plik;

plik.open ("czcionka.psf", ios::in | ios::out | ios::binary);

char t[2000];

plik.read(t, 2000);

// tu wpisujemy nasze instrukcje

// tu plik jest zapisywany z powrotem

plik.seekp(0);

plik.write(t, 2000);

plik.close();

return 0;

}

**Zadanie**

1. Wpisz powyższy program lub ściągnij go ze strony <http://www.mimuw.edu.pl/~amn/io.cpp>
2. Z katalogu /usr/share/consolefonts skopiuj plik z czcionką do bieżącego katalogu: (pamiętaj o tabulatorze!)  
   $ cp /usr/share/consolefonts/Lat2-Fixed13.psf.gz .
3. Rozpakuj go komendą: (pamiętaj o tabulatorze!)

$ gzip –d Lat2-Fixed13.psf.gz

1. Zrób jego kopię o nazwie „czcionka.psf”:

$ cp Lat2-Fixed13.psf czcionka.psf

W pierwszej części program odczytuje pierwsze 2000 bajtów z pliku „czcionka.psf”. Bajty odczytane są do tabeli „t”. Pierwszy bajt przechowywany jest w zmiennej t[0], drugi w t[1], pięćsetny w t[499] a pięćset pierwszy w t[500]. W miejscu wskazanym komentarzem można wpisać dowolne instrukcje zmieniające zawartość tabeli „t”. Ostatnia część programu nagrywa zawartość tabeli z powrotem do pliku.

1. Otwórz plik „czcionka.psf” w edytorze „hte”. Obejrzyj jego zawartość: pierwsze cztery bajty to nagłówek. Następnie kolejno wpisane są kody czcionek, po trzynaście bajtów na czcionkę, w kolejności zgodnej z kodami ASCII. Wiedząc, że litera „s” ma numer 115 w kodach ASCII i wiedząc, że kody zaczynają się od zera, wyznacz w jakim miejscu pliku znajduje się opis czcionki dla litery „s”.
2. Sprawdź, co zrobi z czionką ciąg instrukcji:

n = 1000;

t[n] = t[n] | (t[n] >> 1);

t[n+1] = t[n+1] | (t[n+1] >> 1);

t[n+2] = t[n+2] | (t[n+2] >> 1);

t[n+3] = t[n+3] | (t[n+3] >> 1);

t[n+4] = t[n+4] | (t[n+4] >> 1);

t[n+5] = t[n+5] | (t[n+5] >> 1);

t[n+6] = t[n+6] | (t[n+6] >> 1);

t[n+7] = t[n+7] | (t[n+7] >> 1);

t[n+8] = t[n+8] | (t[n+8] >> 1);

t[n+9] = t[n+9] | (t[n+9] >> 1);

t[n+10] = t[n+10] | (t[n+10] >> 1);

t[n+11] = t[n+11] | (t[n+11] >> 1);

t[n+12] = t[n+12] | (t[n+12] >> 1);

Przetestować jego działanie można używając instrukcji „setfont czcionka.psf”

1. Co stanie się, gdy ten program uruchomisz drugi raz? Trzeci raz?
2. Napisz program, który odwraca czcionkę „s” do góry nogami;
3. pochyla ją;
4. zamienia ją w negatyw.
5. Czy umiałbyś wykonać te operacje dla wszystkich czcionek naraz?