Exercícios complementares - Operadores

1. Escreva um programa que receba um valor em decimal e converta para binário

```c
#include <stdio.h>

int main(){
    int numero, i, aux;

    printf("Digite um número:");
    scanf("%d", &numero);

    printf("%d em binário: ", numero);

    for(i = 7; i>=0; i--){ //A quantidade de casa que irá mostrar é 8
        aux = numero & (0b1 << i); // Mascara o valor apenas no bit de interesse
        aux = aux >> i; // Desloca o bit para início, de forma a mostrar apenas 0 ou 1;
        printf("%d", aux);
    }

    return 0;
}
```

outra solução:

```c
#include <stdio.h>

int main(){
    int numero, i, aux;

    printf("Digite um número:");
    scanf("%d", &numero);

    printf("%d em binário: ", numero);

    for(i = 7; i>=0; i--){ //A quantidade de casa que irá mostrar é 8
        aux = 0b1 << i;
        if(numero >= (aux)){
            printf("%d", 1);
            numero -= aux;
        }
        else{
            printf("%d", 0);
        }

    }

    return 0;
}
```

ou ainda:

```c
#include <stdio.h>

int main(){
    int numero, i;
    int b1 = 0, b2 = 0, b3 = 0, b4 = 0, b5 = 0, b6 = 0, b7 = 0, b8 = 0;

    printf("Digite um número:");
    scanf("%d", &numero);

    if(numero / 128 > 0){
        b8 = 1;
        numero = numero - 128;
    }
    if(numero / 64 > 0){
        b7 = 1;
        numero = numero - 64;
    }
    if(numero / 32 > 0){
        b6 = 1;
        numero = numero - 32;
    }
    if(numero / 16 > 0){
        b5 = 1;
        numero = numero - 16;
    }
    if(numero / 8 > 0){
        b4 = 1;
        numero = numero - 8;
    }
    if(numero / 4 > 0){
        b3 = 1;
        numero = numero - 4;
    }
    if(numero / 2 > 0){
        b2 = 1;
        numero = numero - 2;
    }
    if(numero / 1 > 0){
        b1 = 1;
    }


    printf("\tem binario: %d%d%d%d%d%d%d%d", b8, b7, b6, b5, b4, b3, b2, b1);

    return 0;
}
```

2. Um programa que faça o oposto do exercício anterior, receba um número em binário e converta para decimal.

```c
#include <stdio.h>

int main(){
    unsigned char num = 0b111111;
    int i, soma=0, tmp;

    for(i = 0 ; i<7; i++){
        tmp = (num & (0b1 << i)); // Deixa o valor apenas da casa de interesse;
        soma += tmp;

    }
    printf("Resultado: %d\n", soma);

    return 0;
}
```

```c
#include <stdio.h>

int main(){
    char num[8]; // Armazena o número binário em string
    int i, soma=0, tmp;

    printf("Digite o valor em binário preenchido com 0 ser tiver menos de 8 casas:
\n");
    scanf("%s", num); // Nesse caso não vai o &

    for(i = 0 ; i < 8; i++){
        if(num[i] == '1'){ // Se o valor da casa for igual a 1
            tmp = 0b10000000 >> i; // define o peso da casa
            soma += tmp; // Adiciona os pesos
        }

    }
    printf("Resultado: %d\n", soma);

    return 0;
}
```

3. Faça a função qtdeDeBitsParesLigados(int n) que retorna o número de bits pares que estão ligados.[1]

   Ex:

```
main(){
    qtdeDeBitsParesLigados(73);   //se   73   =   0100   1001,   então
qtdeDeBitsParesLigados == 2
}
```

```c
#include <stdio.h>

int main(){
    int numero, i, aux, count = 0;

    printf("Digite um número:");
    scanf("%d", &numero);
    printf("\t%d em binário: ", numero);

    for(i = 7; i>=0; i--){ // A estrutura é parecida com o programa que converte
para binário.
        aux = numero & (0b1 << i);
        aux = aux >> i;
        printf("%d", aux);
        if( i % 2 == 0 && aux == 1) //Se a casa for par (i % 2 == 1) e o valor da
casa for 1:
            count++;
    }

    printf("\n\tQuantidade de pares ligados: %d\n", count);

    return 0;
}
```

4. Escreva uma função *criptografa(int n)* que recebe um inteiro n com 8 bits (índices: 7,6,5,4,3,2,1,0) e que retorna esse inteiro embaralhando esses bits para a seguinte sequência (7,5,3,1,6,4,2,0)[1]

   Ex:

```
main(){
criptografa(73); // se 73 = 0100 1001, então criptografa(73) == 0010
1001 == 41
}
```

[1] Ensino Superior e Técnico em Informática (UNIBRATEC) - Professor: Frederico Brito Fernandes

5. Faça uma função *pisca(int milissegundos)* que ora liga todos os bits pares e ora liga todos os bits ímpares num intervalo de 2 segundo entre as mudanças, imitando o efeito de um pisca-pisca de natal. Apresente os valores em binários de como ficaria a saída. O valor de milissegundos é usado para fazer uma espera entre a alternância, e para isso use a função *Sleep(<int milissegundos>)* da biblioteca windows.h. [1]

```
#include <stdio.h>
#include <windows.h>

void pisca(int tempo, unsigned char v);
void printbin(unsigned char numero);

int main(){
    unsigned char v = 0b10101010;
    pisca(1000, v);

    return 0;
}

void pisca(int tempo, unsigned char v){
    while(1){
        printbin(v);
        v = ~v ;
        Sleep(tempo);
    }
}

void printbin(unsigned char numero){
    int i;
    unsigned char aux;
    for(i = 7; i>=0; i--){ //A quantidade de casa que irá mostrar é 8
        aux = numero & (0b1 << i); // Mascara o valor apenas no bit de interesse
        aux = aux >> i; // Desloca o bit para início, de forma a mostrar apenas 0
ou 1;
        printf("%d", aux);
    }
    printf("\n");
}
```

6. Faça uma função piscaUmIndoEVoltando(int milissegundos) que faz os leds pares serem ligados sequencialmente, um de cada vez e de forma crescente, e em seguida, os leds ímpares serem ligados de forma descrescente. [1]

7. Faça um programa que leia um byte do teclado e a seguir zere os bits 3 e 4, e inverta os bits 0 e 7. O resultado deverá ser mostrado em hexadecimal na tela.[2]

[2] Operadores Bit-A-Bit - mesquita@cefetsp.br