Operadores a nivel de Bit

Bruno Vega 5º 1º Electronica

Ejercicio Nº1

Escribe una función que multiplique un número entero ingresado por el usuario por 2 utilizando un operador a nivel de bit. Mostrar el resultado por pantalla.

```
#include <stdio.h>
#define BYTE_TO_BINARY_PATTERN "%c%c%c%c%c%c%c%c"
#define BYTE_TO_BINARY(byte)  \
  ((byte) & 0x80 ? '1' : '0'), \
  ((byte) & 0x40 ? '1' : '0'), \
  ((byte) & 0x20 ? '1' : '0'), \
  ((byte) & 0x10 ? '1' : '0'), \
  ((byte) & 0x08 ? '1' : '0'), \
  ((byte) & 0x04 ? '1' : '0'), \
  ((byte) & 0x02 ? '1' : '0'), \
  ((byte) & 0x01 ? '1' : '0')

int multPorDos(int num);

int main() {
    int num;
    printf("Ingrese un número entero (0-255): "); scanf("%d", &num);
    num = multPorDos(num);
    printf("Decimal: %d\n", num);
    printf("Binario: "BYTE_TO_BINARY_PATTERN, BYTE_TO_BINARY(num));
printf("\n");
    return 0;
}

int multPorDos(int num){
    return num << 1;
}
```

Ejercicio Nº2

Escribe una función que divida un número entero ingresado por el usuario por 2 utilizando un operador a nivel de bit. Mostrar el resultado por pantalla.

```
#include <stdio.h>
#define BYTE_TO_BINARY_PATTERN "%c%c%c%c%c%c%c%c"
#define BYTE_TO_BINARY(byte)  \
  ((byte) & 0x80 ? '1' : '0'), \
  ((byte) & 0x40 ? '1' : '0'), \
  ((byte) & 0x20 ? '1' : '0'), \
```

```
  ((byte) & 0x10 ? '1' : '0'), \
  ((byte) & 0x08 ? '1' : '0'), \
  ((byte) & 0x04 ? '1' : '0'), \
  ((byte) & 0x02 ? '1' : '0'), \
  ((byte) & 0x01 ? '1' : '0')

int divPorDos(int num);

int main() {
    int num;
    printf("Ingrese un número entero (0-255): "); scanf("%d", &num);
    num = divPorDos(num);
    printf("Decimal: %d\n", num);
    printf("Binario: "BYTE_TO_BINARY_PATTERN, BYTE_TO_BINARY(num));
printf("\n");
    return 0;
}

int divPorDos(int num){
    return num >> 1;
}
```

Ejercicio N°3

Escribe una función que cuente y retorne la cantidad de bits encendidos (bits con valor 1) en un número entero ingresado por el usuario (pista: ">>" y "&"). Mostrar el resultado por pantalla.

```
#include <stdio.h>

int countSetBits(int num);

int main() {
    int num;
    printf("Ingrese un número entero (0-255): "); scanf("%d", &num);
    printf("Cantidad de bits encendidos (1): %d\n", countSetBits(num));
    return 0;
}

int countSetBits(int num) {
    int count = 0;
    while (num > 0) {
        count += num & 1;
        num = num >> 1;
    }
    return count;
}
```

Ejercicio N°4

Escribe una función que verifique si el bit más significativo de un número entero ingresado por el usuario es 1 o 0 (pista: "&"). Mostrar el resultado por pantalla.

```c
#include <stdio.h>

int checkMSB(int num);

int main() {
    int num;
    printf("Ingrese un número entero (0-255): "); scanf("%d", &num);
    if (checkMSB(num))
        printf("El bit más significativo es 0.\n");
    else
        printf("El bit más significativo es 1.\n");;
    return 0;
}

int checkMSB(int num) {
    int msb = num & 0x80;
    if (msb == 0)
        return 1;
    else
        return 0;
}
```

Ejercicio N°5

Escribe una función que establezca el bit en una posición específica de un número entero ingresado por el usuario (pista: "|"). Mostrar el resultado por pantalla.

```c
#include <stdio.h>

void setBit(int* num, int position);

int main() {
    int num, position;
    printf("Ingrese un número entero (0-255): "); scanf("%d", &num);
    printf("Ingrese la posición del bit (0-7): "); scanf("%d", &position);
    setBit(&num, position);
    printf("Resultado: %d\n", num);
    return 0;
}

void setBit(int* num, int position) {
    int mask = 1 << position;
    *num = *num | mask;
}
```

Ejercicio N°6

Escribe una función que borre el bit en una posición específica de un número entero ingresado por el usuario (pista: "&" y "~"). Mostrar el resultado por pantalla.

```c
#include <stdio.h>

void clearBit(int* num, int position);

int main() {
    int num, position;
    printf("Ingrese un número entero (0-255): "); scanf("%d", &num);
    printf("Ingrese la posición del bit (0-7): "); scanf("%d", &position);
    clearBit(&num, position);
    printf("Resultado: %d\n", num);
    return 0;
}

void clearBit(int* num, int position) {
    int mask = ~(1 << position);
    *num = *num & mask;
}
```

Ejercicio N°7

Escribe una función que realice una rotación circular a la derecha en un número entero ingresado por el usuario utilizando los operadores a nivel de bits.

```c
#include <stdio.h>

void rotateRight(int* num);

int main() {
    int num;

    printf("Ingrese un número entero (0-255): "); scanf("%d", &num);
    rotateRight(&num);
    printf("Resultado: %d\n", num);
    return 0;
}

void rotateRight(int* num) {
    int msb = *num & 0x01;
    *num = (*num >> 1) | (msb << 7);
}
```

Ejercicio N°8

Escribe un programa que invierta los valores de dos variables enteras ingresadas por el usuario, sin utilizar una variable auxiliar, utilizando el operador XOR y la propiedad de que a = a ^ b ^ b cuando a y b son

enteros.

```c
#include <stdio.h>

void swap(int* a, int* b);

int main() {
    int num1, num2;
    printf("Ingrese el primer número: "); scanf("%d", &num1);
    printf("Ingrese el segundo número: "); scanf("%d", &num2);
    printf("Valores originales:     num1 = %d, num2 = %d\n", num1, num2);
    swap(&num1, &num2);
    printf("Valores intercambiados: num1 = %d, num2 = %d\n", num1, num2);
    return 0;
}

void swap(int* a, int* b) {
    *a = *a ^ *b;
    *b = *a ^ *b;
    *a = *a ^ *b;
}
```

Ejercicio Nº9

Escribe una función que intercambie el valor de dos bits en posiciones específicas de un número entero ingresado por el usuario (pista: "^"). Mostrar el resultado por pantalla.

```c
#include <stdio.h>

void swapBits(int* num, int pos1, int pos2);

int main() {
    int num, pos1, pos2;
    printf("Ingrese un número entero (0-255): "); scanf("%d", &num);
    printf("Ingrese la posición del primer bit (0-7): "); scanf("%d",
&pos1);
    printf("Ingrese la posición del segundo bit (0-7): "); scanf("%d",
&pos2);
    printf("Número original: %d\n", num);
    swapBits(&num, pos1, pos2);
    printf("Resultado: %d\n", num);
    return 0;
}

void swapBits(int* num, int pos1, int pos2) {
    int bit1 = (*num >> pos1) & 1;
    int bit2 = (*num >> pos2) & 1;
    if (bit1 != bit2)
        *num ^= (1 << pos1) | (1 << pos2);
}
```

Ejercicio N°10

Escribe una función que verifique si dos números enteros ingresados por el usuario tienen al menos un bit encendido en común (pista: "&"). Mostrar el resultado por pantalla.

```c
#include <stdio.h>

int haveCommonBits(int num1, int num2);

int main() {
    int num1, num2;
    printf("Ingrese el primer número entero (0-255): "); scanf("%d",
&num1);
    printf("Ingrese el segundo número entero (0-255): "); scanf("%d",
&num2);
    int result = haveCommonBits(num1, num2);
    (result) ? printf("Los números tienen al menos un bit encendido en
común.\n") : printf("Los números no tienen bits encendidos en común.\n");
}

int haveCommonBits(int num1, int num2) {
    int commonBits = num1 & num2;
    return commonBits != 0;
}
```

Ejercicio N°11

Escribe una función que realice la operación lógica NAND entre dos números enteros ingresados por el usuario (pista: "&" y "~"). Mostrar el resultado por pantalla.

```c
#include <stdio.h>

#define BYTE_TO_BINARY_PATTERN "%c%c%c%c%c%c%c%c"
#define BYTE_TO_BINARY(byte)  \
  ((byte) & 0x80 ? '1' : '0'), \
  ((byte) & 0x40 ? '1' : '0'), \
  ((byte) & 0x20 ? '1' : '0'), \
  ((byte) & 0x10 ? '1' : '0'), \
  ((byte) & 0x08 ? '1' : '0'), \
  ((byte) & 0x04 ? '1' : '0'), \
  ((byte) & 0x02 ? '1' : '0'), \
  ((byte) & 0x01 ? '1' : '0')

int logicalNAND(int num1, int num2);

int main() {
    int num1, num2;
    printf("Ingrese el primer número entero (0-255): "); scanf("%d",
```

```
&num1);
    printf("Ingrese el segundo número entero (0-255): "); scanf("%d",
&num2);
    printf("Resultado en binario: "BYTE_TO_BINARY_PATTERN,
BYTE_TO_BINARY(logicalNAND(num1, num2))); printf("\n");
    return 0;
}

int logicalNAND(int num1, int num2) {
    int result = ~(num1 & num2);
    return result;
}
```

Ejercicio N°12

Escribe una función que verifique si dos números enteros ingresados por el usuario son iguales utilizando el operador "^" y una comparación adicional con cero. Mostrar el resultado por pantalla.

```
#include <stdio.h>

int areEqual(int num1, int num2);

int main() {
    int num1, num2;
    printf("Ingrese el primer número entero: "); scanf("%d", &num1);
    printf("Ingrese el segundo número entero: "); scanf("%d", &num2);
    areEqual(num1, num2) ? printf("Los números son iguales.\n") :
printf("Los números no son iguales.\n");
    return 0;
}

int areEqual(int num1, int num2) {
    return (num1 ^ num2) == 0;
}
```