Máster en Programación de Videojuegos
Fundamentos Matemáticos
Profesor José María Benito

Práctica: Representación Numérica

Introducción

El objetivo de esta práctica es familiarizarse con los aspectos más importantes de la representación numérica con números binarios.

Notas Generales

Esta práctica está diseñada para ser realizada en clase y no va a ser corregida o calificada. Sin embargo debes intentar completar todos los puntos y aprovechar el tiempo en clase. En caso contrario será motivo para suspender la asignatura.

Creación del proyecto en Visual Studio

1. En Visual Studio:
   1.1. Crea un nuevo proyecto de consola y dale un nombre estilo “Representacion”.

Ejercicios

1. Añade un fichero common.cpp y common.h. Crea dos funciones complementarias llamadas:

```
int intFromBinary(const std::string& number);

std::string binaryFromInt(int x);
```

La primera función debe aceptar un string de un número binario (tipo “1010”) y devolver el entero equivalente.

La segunda función debe aceptar un número entero e imprimir su representación binaria seguida de retorno de carro "\n".

Después podrás imprimir números enteros en base10 de la siguiente manera:

```
void printIntAsBinary(int x)
{
    std::string str = binaryFromInt(x);
    printf("%s\n", str.c_str());
}
```

Guarda los ejercicios en ficheros con el nombre ejercicioN.cpp y ejercicioN.h, donde N es el número del ejercicio.

1. Utiliza las funciones anteriores para obtener el número entero en base10 correspondiente al número 1010 base 2. ¿A qué número corresponde en base10?
2. Obtén un número entero con el valor 1010 base2.
   2.1. Aplícale el operador NOT (~). Indica los número en base2 y base10 antes y después del operador. Usa la función binaryFromInt y printf.
   2.2. Aplícale el operador << con varios parámetros (p.ej. <<1, <<2, <<5, <<6). Imprime el resultado en binario.
   2.3. Obtén un entero con signo y sin signo a partir del número de 32 bits con todos los bits a uno. Utiliza la función intFromBinary con "111…1". Imprime el valor del numero entero int y unsigned obtenido mediante:

   ```
   printf("signed=%d\n",number);
   printf("unsigned=%d\n",unsignedNumber);
   ```

      2.3.1. ¿Haz un desplazamiento hacia la izquierda (<<) y la derecha a ambos números con signo y sin signo, qué sucede en cada caso?
      2.3.2. Prueba a sumar una unidad al número anterior con signo. Prueba a restar una unidad al número 0 sin signo.
      2.3.3. Utiliza los desplazamientos y el operador & para extraer los canales RGB del siguiente valor:

   ```
   int pink = 0xCC6699;
   ```

3. ¿Qué numero obtienes … ?
   3.1. Al hacer la raíz cuadrada de -1.
   3.2. Al dividir 0 (int) entre int zero = 0;
   3.3. Al dividir 1.0f entre 0.0f.
4. Obtén el módulo 1, 10 y 1000 de 456
   4.1. Escribe un bucle for donde i va de 0 a 10 y obtén el módulo 2 de i.

5. Imprime los siguientes números en base2:

```
int a=4;
int b=-4;
```

5.1. Súmalos e imprime el resultado en base2.

5.2. Justifica los números obtenidos

5.3. Modifica binaryFromInt para imprimir enteros de tipo char.

5.4. Imprime los siguientes números en base2

```
char c = (char)255;
unsigned char uc = (unsigned char)255;
```

5.5. Justifica los números obtenidos

5.6. Imprime los siguientes números en base2

```
char d = (char)128; // only to 127
char e = (char)-128;
```

5.7. Justifica los números obtenidos.

6. Números Coma Flotante (Floating Point.

6.1. Imprime el número float point2 = 0.2f con precisión de 16 cifras mediante:

```
printf("%1.16f\n",point2);
```

6.2. ¿Qué observas?

6.3. A el siguiente número:

```
float f = 0.1f;
```

6.4. hazle las siguientes operaciones equivalentes:

```
float sum = 0;
for (int i = 0; i < 100; ++i)
    sum += f;

float product = f * 100;
```

6.5. Compara sum y product con 15 cifras de precisión. Observa si son iguales resultados y justifica por qué.

6.6. ¿Cómo podrías comparar si son iguales mediante una sentencia if?