

# ESTRUCTURA DE COMPUTADORES

GRADO EN INGENIERÍA INFORMÁTICA



UNIVERSIDAD CARLOS III DE MADRID

## **Práctica 1**

### **Representación de la Información**

Septiembre 2013

## Contenido

<b>OBJETIVOS DE LA PRÁCTICA .....</b>	<b>2</b>
<b>EJERCICIO 1.....</b>	<b>3</b>
<b>EJERCICIO 2.....</b>	<b>4</b>
<b>EJERCICIO 3.....</b>	<b>5</b>
<b>EJERCICIO 4.....</b>	<b>6</b>
<b>EJERCICIO 5.....</b>	<b>7</b>
<b>PROCEDIMIENTO DE EVALUACIÓN DE LA PRÁCTICA .....</b>	<b>8</b>
<b>PROCEDIMIENTO DE ENTREGA DE LA PRÁCTICA .....</b>	<b>9</b>
<b>NORMAS .....</b>	<b>10</b>

## Objetivos de la práctica

El objetivo de la práctica consiste en entender el formato de representación de los números en el computador. Para ello se propone el desarrollo de una serie de programas en C que permiten trabajar con los detalles relacionados con los números enteros y los números en coma flotante representados en el estándar IEEE 754, que es el que se emplea en la mayoría de los lenguajes de programación.

Para el desarrollo de la práctica es necesario que el alumno repase:

- Los diferentes operadores que ofrece C para trabajar con bits. En el siguiente enlace se puede obtener información sobre los diferentes operadores de bit de C.

[http://en.wikipedia.org/wiki/Bitwise\\_operations\\_in\\_C](http://en.wikipedia.org/wiki/Bitwise_operations_in_C)

- La representación de números en coma flotante según el estándar IEEE754 de precisión simple.

La práctica consistirá en realizar los siguientes ejercicios

- Ejercicio 1: analizar el ancho de los tipos de datos *float*, *double*, *int* y *long*.
- Ejercicio 2: obtener el valor de un número binario en complemento a 2.
- Ejercicio 3: obtener el valor de un número binario en coma flotante según el estándar IEEE754.
- Ejercicio 4: Sumar dos números binarios en complemento a 2 y obtener el valor de la suma.
- Ejercicio 5: obtener el siguiente número representable dado un número en coma flotante según el estándar IEEE754.

## Ejercicio 1.

- a. Escriba un programa en C que calcule la siguiente serie:

$$S_n = 1 + \frac{1}{2} + \frac{1}{2^2} + \frac{1}{2^3} + \frac{1}{2^4} + \cdots + \frac{1}{2^n}$$

Para ello se deben utilizar números en coma flotante de precisión simple (variables de tipo *float*) y números en coma flotante de precisión doble (variables de tipo *double*). El objetivo del programa es calcular la serie e indicar el número de iteraciones a partir del cual el resultado que se obtiene para  $S_{n+1}$  es igual que el obtenido para  $S_n$ .

- b. Agregue una función al anterior programa que calcule el tamaño de una variable de tipo entero (variable de tipo *int*) y una variable de tipo entero largo (variable de tipo *long*). Para ello, el alumno programará un bucle en el cual la variable tendrá un valor inicial de 0 e irá incrementándose. En el momento en que la variable se desborde y pase a tener un valor negativo, el alumno deberá imprimir su valor y analizarlo para obtener cuantos bits forman el ancho del tipo de datos.

Documentación a entregar:

- **E1-Series.c.** Contiene el código fuente del programa conteniendo ambas funciones. El programa debe contener documentación describiendo el funcionamiento, así como la versión y el autor o autores del mismo. Para la realización de esta parte se usará como base el código entregado, donde vienen especificados los nombres de los métodos y el retorno de los mismos.
- Además, se responderá en un apartado específico de la **memoria** a las siguientes preguntas:
  - ¿Cuál es el valor máximo de iteraciones con precisión simple?
  - ¿Cuál es el valor máximo de iteraciones con precisión doble?
  - ¿Cuál es el tamaño en bits del tipo *int*?
  - ¿Cuál es el tamaño en bits del tipo *long*?
  - ¿Qué ocurre con la precisión si en lugar de utilizar una máquina de 32 bits pasamos a una de 64 bits, o viceversa? Justifique su respuesta

## Ejercicio 2.

Escriba un programa en C denominado *E2-CompToInt.c*, utilizando como base el código fuente entregado, que dado un número **binario en complemento a 2 de 8 bits** muestre su valor decimal. Para ello se debe rellenar la función `comp2ToInt` que se encuentra dentro del código base.

- `int comp2ToInt(bit* num)`

Función que recibe como parámetro un array de bits que representan un número binario en complemento a dos y retorna su valor entero.

Documentación a entregar:

- **E2-CompToInt.c.** Programa que se entrega como código fuente con la función `comp2ToInt` implementada. Se pueden agregar al programa funciones adicionales si se considera oportuno. El código debe contener documentación describiendo su funcionamiento, así como el nombre del o de los autores.

### NOTAS:

- Considere  $n=8$  bits para hacer el complemento a 2.
- Se deben respetar los nombres asignados, así como los retornos especificados en el programa *E2-CompToInt.c* que se entrega.
- No se pueden emplear funciones del API de C.

### Ejercicio 3.

Escriba un programa en C denominado *E3-IEEE754ToFloat.c*, utilizando como plantilla el programa entregado junto con la documentación, que recibe un número binario que representa un número en coma flotante según el estándar IEEE 754 de 32 bits y retorne su valor. La cadena que recibe el programa puede tener separados por un espacio la representación del signo, la representación del exponente y la representación de la mantisa de la siguiente forma:

```
$ ./E3-IEEE754ToFloat 1 10100001 10101110100111111000000
```

También puede introducirse el número en un solo argumento:

```
$ ./E3-IEEE754ToFloat 110100001101011110100111111000000
```

Para ello se debe rellenar la función `IEEE754ToFloat` que se encuentra dentro del código base.

- `float IEEE754ToFloat (bit* numero)`  
Función que recibe como parámetro un array de bits que representan un número en coma flotante según el estándar IEEE754 de precisión simple y devuelve su valor decimal  
Ejemplo:

Array de bits IEEE754	Valor
0 10000011 10011001100110011001101	+25.6
1 10000100 10101010000000000000000	-53.25

Además se responderá en un apartado específico de la **memoria** a la siguiente pregunta:

- a) Calcular el número de números representables que hay entre 8 y 9, y entre 15 y 16. ¿Dónde hay un número mayor de números representables? Justifique su respuesta.

Documentación a entregar:

- **E3-IEEE754ToFloat.c:** Programa que se entrega como código fuente con la función `IEEE754ToFloat` implementada. Se pueden agregar al programa funciones adicionales si se considera oportuno. El código debe contener documentación describiendo su funcionamiento, así como el nombre del o de los autores.

#### NOTAS:

- Se deben respetar los nombres asignados, así como los retornos especificados en el programa *E3-IEEE754ToFloat.c* que se entrega.
- No se pueden funciones del API de C salvo la función *pow* que calcula la potencia de un número *x* elevado a un número *y*. Ver *man pow* para más información.
- Se deben tener en cuenta los casos particulares del estándar: NaN,  $\pm\infty$  y números no normalizados.

## Ejercicio 4.

Escriba un programa en C denominado *E4-Add.c*, utilizando como plantilla el programa entregado junto con la documentación, que reciba dos números **binarios en complemento a 2 de 8 bits**, realice la suma bit a bit, y una vez obtenido el resultado en binario, obtenga su valor decimal de la misma manera que en el ejercicio 2. Para ello se debe rellenar la función `add` que se encuentra dentro del código base.

- `int add(bit* num1, bit* num2)`  
Función que recibe como parámetro dos array de bits que representan dos números binarios en complemento a dos y retorna el valor entero de la suma de ambos.

Documentación a entregar:

- **E4-Add.c:** Programa que se entrega como código fuente con la función `add` implementada. Se pueden agregar al programa funciones adicionales si se considera oportuno. El código debe contener documentación describiendo su funcionamiento, así como el nombre del o de los autores.

### NOTAS:

- Si se desea emplear código o funciones desarrollados en el Ejercicio 2, estos deben ser copiados al programa *E4-Add.c*
- Se deben respetar los nombres asignados, así como los retornos especificados en el programa *E4-Add.c* que se entrega.
- No se pueden funciones del API de C.
- Se deberá realizar en primer lugar la suma bit a bit de los números binarios, y a partir del número resultante, obtener el valor decimal en un segundo paso.
- En caso de desbordamiento el bit de acarreo será descartado.

## Ejercicio 5.

Escriba un programa en C denominado *E5-NextUp.c*, utilizando como plantilla el programa entregado junto con la documentación, que reciba un número binario que representa un número en coma flotante según el estándar IEEE 754 de 32 bits. devuelva el siguiente número representable. En caso de que el número sea 0, el método debe devolver el primer número no normalizado. En caso de que el número sea el mayor número normalizado, el programa debe devolver Infinity. Se deberá imprimir el valor decimal tanto del número como de su siguiente (puede utilizar para ello las funciones desarrolladas en el ejercicio 3). Se debe rellenar la función `nextup` que se encuentra dentro del código base.

- `void nextup(bit* num, bit* nextnum)`

Función que recibe dos parámetros. El primero es un array de bits que contiene el número a partir del cual se va a calcular el siguiente representable. El segundo es un array donde se va a almacenar el siguiente representable calculado.

Documentación a entregar:

- **E5-NextUp.c:** Programa que se entrega como código fuente con la función `nextup` implementada. Se pueden agregar al programa funciones adicionales si se considera oportuno. El código debe contener documentación describiendo su funcionamiento, así como el nombre del o de los autores.

### NOTAS:

- Se deben respetar los nombres asignados, así como los retornos especificados en el programa *E5-NextUp.c* que se entrega.
- No se pueden funciones del API de C salvo la función *pow* que calcula la potencia de un número *x* elevado a un número *y*. Ver *man pow* para más información.
- Se deben tener en cuenta los casos particulares del estándar: NaN,  $\pm\infty$  y números no normalizados.
- Si se desea emplear código o funciones desarrollados en el Ejercicio 3, estos deben ser copiados al programa *E5-NextUp.c*

## Procedimiento de evaluación de la práctica

La evaluación de la práctica se va a dividir en dos partes.

- **Código (8 puntos)**
  - Ejercicio 1 (*1 puntos*)
  - Ejercicio 2 (*1.5 puntos*)
  - Ejercicio 3 (*1.5 puntos*)
  - Ejercicio 4 (*2 puntos*)
  - Ejercicio 5 (*2 puntos*)
- **Memoria (2 puntos)**

Los ejercicios 1, 2, 3, 4 y la memoria son obligatorios. Deben entregarse para poder seguir la evaluación continua.

## Procedimiento de entrega de la práctica

La entrega de la práctica 1 se realizará de la siguiente manera: se habilitarán dos entregadores distintos, uno para los **ejercicios** de este cuaderno y otro para la memoria completa. La fecha límite de entrega es el día **25 de Octubre de 2013 a las 23:55 horas**.

**Entregador 1:** Se deberá entregar un único archivo comprimido en formato zip con el nombre `ec_p1_AAAAAAAAAA_BBBBBBBBBB.zip` donde A...A y B...B son los NIAs de los integrantes del grupo.

El archivo zip debe contener:

- **Bit-Library.c**
- **Bit-Library.h**
- **E1-Series.c**
- **E2-CompToInt.c**
- **E3-IEEE754ToFloat.c**
- **E4-Add.c**
- **E5-NextUp.c**
- **Makefile**

**Entregador 2:** Se deberá entregar la memoria en un único archivo en formato pdf con el nombre `ec_p1_AAAAAAAAAA_BBBBBBBBBB.pdf` donde A...A y B...B son los NIAs de los integrantes del grupo. La memoria tendrá que contener al menos los siguientes apartados:

- Portada donde figuren los autores (incluyendo nombre completo, NIA y dirección de correo electrónico).
- Índice de contenidos.
- Descripción de todos los programas C solicitados en este cuaderno de prácticas detallando las principales funciones implementadas. La memoria debe describir el comportamiento de los programas, así como las principales decisiones de diseño (adicionalmente se pueden incluir diagramas de flujo, algoritmos, etc.).
- Las respuestas a las preguntas planteadas en los ejercicios debidamente justificadas.
- Batería de pruebas utilizadas y resultados obtenidos. Se dará mayor puntuación a pruebas avanzadas, casos extremos, y en general a aquellas pruebas que garanticen el correcto funcionamiento de la práctica en todos los casos. Hay que tener en cuenta:
  - Que un programa compile correctamente y sin advertencias (warnings) no es garantía de que funcione correctamente.
  - Evite pruebas duplicadas que evalúan los mismos flujos de programa. La puntuación de este apartado no se mide en función del número de

pruebas, sino del grado de cobertura de las mismas. Es mejor pocas pruebas que evalúan diferentes casos a muchas pruebas que evalúan siempre el mismo caso.

- Conclusiones y problemas encontrados.

**NOTA: NO DESCUIDE LA CALIDAD DE LA MEMORIA DE SU PRÁCTICA.**

Aprobar la memoria es tan imprescindible para aprobar la práctica, como el correcto funcionamiento de la misma. Si al evaluarse la memoria de su práctica, se considera que no alcanza el mínimo admisible, su práctica estará suspensa.

**La longitud de la memoria no deberá superar las 10 páginas** (portada e índice incluidos)

- La entrega de las prácticas ha de realizarse de forma electrónica. En AULA GLOBAL se habilitarán unos enlaces a través de los cuales podrá realizar la entrega de las prácticas.
- La única versión registrada de su práctica es la última entregada. La valoración de esta es la única válida y definitiva.

## **Normas**

- 1) Las prácticas que no compilen o que no se ajusten a la funcionalidad y requisitos planteados, obtendrán una calificación de 0.
- 2) Un programa no comentado, obtendrá una calificación de 0.
- 3) La entrega de la práctica se realizará a través de los entregadores habilitados. No se permite la entrega a través de correo electrónico sin autorización previa.
- 4) Se prestará especial atención a detectar funcionalidades copiadas entre dos prácticas. En caso de encontrar implementaciones comunes en dos prácticas, ambas obtendrán una calificación de 0.