ESTRUCTURA DE COMPUTADORES

GRADO EN INGENIERÍA INFORMÁTICA

UNIVERSIDAD CARLOS III DE MADRID

Grupo de Arquitectura de Computadores

Práctica 1
Representación de la Información

Curso 2015/2016

Contenido

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 lenguaje 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 computadores y lenguajes de programación.

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

- Los principales aspectos del lenguaje de programación C:

  Material disponible en aula global.

- 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

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

Ejercicio 1.

Para cada uno de los tipos de datos *char, int,unsigned int, long, unsingned long, float* y *double* del lenguaje C, escriba un programa en C que únicamente muestre por pantalla el nombre del tipo, su tamaño en bits, así como el menor y mayor número representable (rango desde el menor negativo hasta el mayor positivo).

Código a entregar:

- Archivo p1-e1.c con el código pedido.
- El programa no tendrá parámetros e imprimirá la información pedida utilizando el siguiente formato:

```
char  Tamaño  Menor  Mayor
int   Tamaño  Menor  Mayor
unsigned int   Tamaño   Menor   Mayor
long   Tamaño   Menor   Mayor
unsigned long    Tamaño    Menor    Mayor
float Tamaño  Menor_positivo_no_normalizado  Menor_positivo_normalizado  Mayor
double Tamaño Menor_positivo_no_normalizado  Menor_positivo_normalizado  Mayor
```

Se usará el modificador %G, en la función *printf*, para imprimir los valores de tipo *float* y *double*, y el que que corresponda para el resto.

Para obtener el valor mayor y menor, se deberá investigar y encontrar aquellas **constantes** definidas en el último estándar de C (c11) que permiten conocer dichos valores.

Para forzar al compilador a utilizar este estándar se deberá compilar de la siguiente forma:

```
gcc –Wall –Werror  –std=c11  –o  p1-e1   p1-e1.c
```

Documentación a entregar del ejercicio en la memoria:

- Código fuente del programa realizado por su grupo de prácticas que correctamente realiza lo pedido.
- Pruebas realizadas que estime oportuno presentar.
- Salida del programa tras su ejecución en la máquina *guernika.lab.inf.uc3m.es*
- Justificación breve de los resultados obtenidos.

  Esto puede incluir revisar si son coherentes (si un *int* tiene 8 bits no es coherente que el valor máximo representable sea 1024), e interpretar estos resultados con la teoría dada (indicar si el rango del tipo *int* supone que es en complemento a 1 o a 2 la representación usada).

Ejercicio 2.

Escriba un programa en C que imprima el resultado de ejecutar el siguiente fragmento de código:

```
int i;
float r = 0.0;
for (i=0; i<10; i++)
        r = r + 0.1;
```

Es decir, imprime el resultado de la expresión 10 x 0.1 implementada como un bucle que suma 0.1 diez veces.

Código a entregar:

- Archivo p1-e2.c con el código pedido.
- El programa no tendrá parámetros e imprime la información pedida con el siguiente formato:

  `resultado error`

  Siendo *resultado* el valor de r tras la ejecución del bucle y *error*, el error cometido. Se usará la función *printf* utilizando el modificador %.10f (imprime el valor con 10 dígitos decimales) para ambos valores (resultado y error). Para almacenar el valor de r se utilizará una variable de tipo *float*.

Documentación a entregar del ejercicio en la memoria:

- Código fuente de los programas realizados por su grupo de prácticas que correctamente realiza lo pedido.
- Pruebas realizadas que estime oportuno presentar.
- Salida de los programas tras su ejecución en la máquina *guernika.lab.inf.uc3m.es.*
- Justificación brevemente de los resultados obtenidos.

Ejercicio 3.

a. Escriba un programa en C que calcule el valor de la serie:

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

Usando números en coma flotante de precisión simple (variables de tipo *float*) y únicamente imprima el valor de la serie y 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. Es decir, el valor de n y de S_n que cumple esa condición.

b. Escriba un programa en C que calcule el valor de la serie:

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

Usando números en coma flotante de precisión doble (variables de tipo *double*) y únicamente imprima el valor de la serie y 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. Es decir, un segundo programa que en lugar de usar tipo *float* use un tipo *double*.

Código a entregar:

- Archivos p1-e3a.c y p1-e3b.c con el código pedido.
- El programa no tendrá parámetros e imprime la información pedida con el siguiente formato:

  ```
  Tipo n  Sn
  ```

  Siendo Tipo bien *float* o *double*, según corresponda. Se usará el modificador %G para imprimir Sn.

Documentación a entregar del ejercicio en la memoria:

- Código fuente de los programas realizados por su grupo de prácticas que correctamente realiza lo pedido.
- Pruebas realizadas que estime oportuno presentar.
- Salida de los programas tras su ejecución en la máquina *guernika.lab.inf.uc3m.es*
- Justificación brevemente de los resultados obtenidos.

Ejercicio 4.

Escriba un programa en C que recibe como parámetro un número en coma flotante (*float*) y retorna su representación en binario según el estándar IEEE 754 de 32 bits.

El programa se ejecutará de la siguiente forma:

```
$ ./p1-e4 25.6
0   10000011   10011001100110011001101
```

Correspondiendo las tres partes anteriores impresas al signo, exponente y mantisa que están almacenadas en la variable de tipo *float* que sigue el estándar IEEE 754 de 32 bits. Tenga en cuenta que la mantisa almacenada no tiene el bit implícito, y que este no debe mostrarse.

Se deben tener en cuenta todos los casos particulares del estándar. El valor NaN se introduce directamente con NaN (como argumento del programa), el de más infinito se introduce en la entrada (como argumento del programa) utilizando Inf, y el de menos infinito como –Inf.

Código a entregar:

- Archivo p1-e4.c con el código pedido.
- El programa tendrá solo un parámetro e imprime la información pedida con el siguiente formato:

  ```
  Signo Exponente Mantisa
  ```

Documentación a entregar del ejercicio en la memoria:

- Código fuente de los programas realizados por su grupo de prácticas que correctamente realiza lo pedido.
- Pruebas realizadas que estime oportuno presentar.
- Salida de los programas tras su ejecución en la máquina *guernika.lab.inf.uc3m.es*
- Justificación brevemente de los resultados obtenidos en las pruebas.

Ejercicio 5.

Escriba un programa en C que recibe como parámetro un número en coma flotante en binario según el estándar IEEE 754 de 32 bits e imprime su representación como *float*.

El programa se ejecutará de la siguiente forma:

```
$ ./p1-e5 0  10000011  10011001100110011001101
25.6
```

El programa recibe **tres** argumentos como parámetros correspondientes al signo, exponente y mantisa siguiendo el estándar IEEE 754 de 32 bits. El programa imprimirá el número correspondiente.

Se deben tener en cuenta los casos particulares del estándar, tal y como se describió en el ejercicio anterior.

Código a entregar:

- Archivo p1-e5.c con el código pedido.
- El programa tendrá solo tres parámetros e imprime la información pedida con el siguiente formato:

  ```
  Resultado
  ```

  Siendo *Resultado* el valor impreso usando la función *printf* con el modificador %G.

Documentación a entregar del ejercicio en la memoria:

- Código fuente de los programas realizados por su grupo de prácticas que correctamente realiza lo pedido.
- Pruebas realizadas que estime oportuno presentar.
- Salida de los programas tras su ejecución en la máquina *guernika.lab.inf.uc3m.es*
- Justificación brevemente de los resultados obtenidos en las pruebas.

Ejercicio 6.

Escriba un programa en C que recibe como parámetro dos números enteros a y b de tipo *int*, y determine cuántos números enteros comprendidos entre a y b (incluidos a y b) se pueden representar de forma exacta utilizando el estándar IEEE 754 de 32 bits (variables de tipo *float*)

El programa se ejecutará de la siguiente forma:

```
$ ./p1-e6 10000 10005
6
```

```
$ ./p1-e6 100000000 100000005
1
```

Código a entregar:

- Archivo p1-e6.c con el código pedido.
- El programa tendrá solo dos parámetros (valores a y b) e imprime la información pedida con el siguiente formato:

  `Resultado`

  Siendo *Resultado* el valor impreso con la función *printf* utilizando el modificador %d.

Documentación a entregar del ejercicio en la memoria:

- Código fuente de los programas realizados por su grupo de prácticas que correctamente realiza lo pedido.
- Pruebas realizadas que estime oportuno presentar.
- Salida de los programas tras su ejecución en la máquina *guernika.lab.inf.uc3m.es*
- Justificación brevemente de los resultados obtenidos en las pruebas.

Aspectos importantes a tener en cuenta

Normas generales

1) La entrega de la práctica se realizará a través de los entregadores habilitados. No se permite la entrega a través de correo electrónico.
2) La entrega se realizará en el plazo dado por los entregadores. Es posible que para un entregador de Aula Global el fin del plazo para una entrega a las 24:00 termine 10 minutos antes. Revise el soporte de Aula Global.
3) Se prestará especial atención a detectar funcionalidades copiadas entre dos prácticas. En caso de encontrar implementaciones comunes en dos prácticas (o contenidos similares en la memoria), ambas obtendrán una calificación de 0.

Códigos de la práctica

1) Todos los ejercicios se tienen que compilar con los flags –Wall y –Werror del compilador gcc. El último flag convierte todos los avisos y advertencias en error de compilación. Es decir, de la siguiente forma:

```
gcc –Wall  –Werror   -o <nombre del ejecutable>   <nombre del programa (.c)>
```

2) Todos los ejercicios deben seguir el formato de salida que se pide en cada enunciado.
3) Los ejercicios que no compilen o que no se ajusten a la funcionalidad y requisitos planteados, obtendrán una calificación de 0.
    - Esto incluye no usar los nombres de variables, funciones o ficheros pedidos.
4) Un programa no comentado, obtendrá una calificación de 0.
5) Hay que tener en cuenta que un programa que compile correctamente y sin advertencias (*warnings*) no es garantía de que funcione correctamente. Por ello tendrá que realizar aquellas pruebas que garanticen el correcto funcionamiento de la práctica.
6) Todos los ejercicios han de funcionar en **guernika.lab.inf.uc3m.es** (sistema operativo Linux) con el entorno de desarrollo allí disponible. Se utilizará el compilador **gcc disponible** en este sistema. La versión del lenguaje C a usar será C11 (formalmente conocido como ISO/IEC 9899:2011). Esto quiere decir que todos los alumnos deben asegurarse de que sus ejercicios compilan y ejecutan correctamente en esta máquina, independientemente de que puedan funcionar en otros entornos. Para compilar cada uno de los ejercicios tendrá utilizar el siguiente mandato:

```
gcc –Wall –Werror –std=c11 –o   prog   prog.c
```

que permite compilar el programa *prog.c* y generar el fichero ejecutable de nombre *prog*.

Memoria de la práctica

1) La memoria (un único documento) 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.
    - Contenidos pedidos en los distintos ejercicios (una sección por ejercicio).
    - Conclusiones y problemas encontrados.
2) **La longitud de la memoria no deberá superar las 15 páginas** (portada e índice incluidos).
3) Al respecto de la posible descripción de los programas pedidos:
    - Se ha de detallar 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.).
    - Se ha de incluir la 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.
        - 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.

<u>NOTA:</u> 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.

Procedimiento de entrega de la práctica

La entrega de la práctica 1 se realizará de forma electrónica a través de Aula Global

Se habilitarán dos entregadores distintos, uno para el código de los **ejercicios** de este cuaderno y otro para la **memoria** completa. Para ello se habilitará dos enlaces, uno por cada entregador.

La fecha límite de entrega para ambos es el día **18 de cctubre de 2015 a las 23:55 horas.**

Es posible entregar tantas veces como quiera dentro del plazo dado, la única versión registrada de su práctica es la última entregada. La valoración de la práctica es la valoración del contenido de esta última entrega. Revise siempre lo que entregue.

Entregador 1: Se deberá entregar un único archivo comprimido en formato **zip** con el nombre ec_p1_AAAAAAAAA_BBBBBBBBB.zip donde A…A y B…B son los NIA de los integrantes del grupo.

El archivo **zip** debe contener solo los siguientes archivos:

- **p1-e1.c**
- **p1-e2.c**
- **p1-e3a.c**
- **p1-e3b.c**
- **p1-e4.c**
- **p1-e5.c**
- **p1-e6.c**

Entregador 2: Se deberá entregar la memoria en **un único archivo en formato PDF** con el nombre ec_p1_AAAAAAAAA_BBBBBBBBB.pdf donde A…A y B…B son los NIA de los integrantes del grupo.

Evaluación de la práctica

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

- **Código (5 puntos)**
- **Memoria (5 puntos)**

La puntuación de cada ejercicio es proporcional a su dificultad:

- Ejercicio 1 (1 *puntos*)
- Ejercicio 2 (1 *punto*)
- Ejercicio 3 (2 *puntos*)
- Ejercicio 4 (2 *puntos*)
- Ejercicio 5 (*2 puntos*)
- Ejercicio 6 (*2 puntos*)

Si un ejercicio no se entrega su puntuación será 0. Tenga en cuenta que para seguir el proceso de evaluación continua, la nota mínima obtenida en cada práctica debe ser de 2 y la media de las tres prácticas 4.

NOTAS:

1. **Si se detecta un error de concepto grave en la práctica (en cualquier apartado de cualquier ejercicio), la valoración global de toda la práctica será de cero puntos (0 puntos).**
2. **En caso de encontrar implementaciones comunes en dos prácticas (o contenidos similares en la memoria), ambas obtendrán una calificación de 0.**