**Fundamentos de Programación**

**(Grados en Ingeniería Mecánica, Electrónica Industrial y Química Industrial)**

**20 de junio de 2011**

***Ejercicio 1 (4.0 p):*** construir un programa en C lo más modular posible (atendiendo a los criterios de modularidad) y documentar el diseño preliminar con el diagrama de módulos (estructura del programa), la definición en C de las nuevas tipologías de datos y los prototipos de las funciones en C, y el diseño detallado con las definiciones de las respectivas funciones en C.

***Ejercicio 2 (6.0 p):*** implementar en C solamente lo que se le indica en el enunciado (no hay que construir el programa completo).

**Ejercicio 1:** Construir un programa en C para resolver de forma aproximada un sistema lineal de ***n*** ecuaciones con ***n*** incógnitas (***n<100***) mediante el método de Jacobi. Dicho método consta de los siguientes pasos para resolver el sistema lineal especificado a continuación:



1. Se parte de una solución inicial aproximada (denominada semilla). Por ejemplo:



1. Se ejecutan una serie de cálculos para obtener una mejor solución aproximada partiendo de la aproximación semilla inicial. La fórmula que permite construir la siguiente aproximación usando otra se conoce como ecuación de recurrencia, en nuestro caso se obtiene despejando de cada ecuación una incógnita diferente:



1. Se repite el paso anterior pero usando como semilla la última aproximación obtenida, hasta que se cumpla una determinada condición de parada, por ejemplo que la distancia máxima entre las soluciones de dos iteraciones consecutivas sea menor que un valor de precisión prefijado (PREC=0.001):



Una condición que garantiza la convergencia del método de Jacobi es que la matriz de coeficientes original sea diagonalmente dominante, esto es, en cada renglón el valor absoluto del elemento de la diagonal principal es mayor que la suma de los valores absolutos de los elementos restantes del mismo renglón.

El programa leerá los datos del sistema lineal de ecuaciones de un archivo de texto (“sistema.txt”), donde se ha registrado en primer lugar el nº de ecuaciones (1ª línea del archivo) seguido de los coeficientes de las incógnitas y los términos independientes de cada ecuación (almacenados en una línea separada para cada ecuación). A continuación el programa comprobará si la matriz de coeficientes es diagonalmente dominante, en cuyo caso se calculará una solución aproximada y se presentará en pantalla; en caso contrario, se escribirá en pantalla un mensaje indicando que no está garantizada la convergencia a la solución. Ejemplo:

**Sistema a resolver:**

10 x1 - x3 = -1

4 x1 + 12 x2 - 4 x3 = 8

4 x1 + 4 x2 + 10 x3 = 4

**Ecuaciones de recurrencia:**

x1= 0.100 x3 - 0.100

x2= -0.333 x1 + 0.333 x3+ 0.667

x3= -0.400 x1 - 0.400 x2 +0.400

***“sistema.txt”***

3

10 0 -1 -1

4 12 -4 8

4 4 10 4

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | **Soluciones aproximadas en cada iteración** | | | | | | |
| **0** | **1** | **2** | **3** | **…** | **10** | **11** |
| **x1** | 1.000 | 0.200 | -0.180 | -0.121 |  | -0.086 | -0.086 |
| **x2** | 2.000 | 1.333 | 0.333 | 0.656 |  | 0.741 | 0.741 |
| **x3** | 3.000 | -0.800 | -0.213 | 0.339 |  | 0.138 | 0.138 |
| **dmax** |  | 3.800 | 1.000 | 0.552 |  | 0.001 | 0.000 |

**Ejercicio 2:** Diseñar e implementar en C una estructura de datos para representar una lista de ecuaciones químicas (hasta un máximo de ***100***). Nota: una ecuación química representa una reacción química en la que intervienen una serie de sustancias o fórmulas moleculares, que se subdividen en reactivos (parte izquierda de la ecuación) y productos (parte derecha), las cuales van precedidas por unos números (coeficientes estequiométricos) que indican cuantas moléculas de cada sustancia intervienen en la reacción. Cada fórmula molecular indica los elementos presentes en una sustancia (mediante su símbolo atómico) así como la cantidad exacta de átomos de cada elemento en una molécula de la misma. Ejemplos:

2 C8H18 + 25 O2 🡪 16 CO2 + 18 H20

2 H2O 🡪 2 H2 + O2

Considerar solamente ecuaciones químicas en las que intervienen un máximo de 5 reactivos y 5 productos, y donde cada fórmula molecular tiene un máximo de 5 elementos atómicos. Los símbolos de los respectivos elementos atómicos se encuentran almacenados en un archivo binario con la siguiente estructura:

***Elementos\_TP.bin***

Núm. atómico (nº protones): entero (2 bytes)

Nº másico (protones+neutrones): real (simple precisión)

Símbolo: cadena de 3 caracteres (incluye ‘\0’)

Nombre: cadena de 15 caracteres (incluye ‘\0’)

Dicho archivo incluye los 108 elementos de la tabla periódica y se encuentra clasificado alfabéticamente por el símbolo del elemento. Implementar en C los prototipos y las definiciones de las siguientes funciones:

1. Una función que inicialice un vector de cadenas de hasta 3 caracteres con los símbolos de los 108 elementos registrados en el archivo ***elementos\_TP.bin***.
2. Una función lo más eficiente posible que localice y devuelva a través de su identificador la posición en el vector de elementos inicializado en la función anterior de un elemento dado su símbolo como argumento (-1 en caso de no encontrarse).
3. Una función que lea por teclado una fórmula válida y la devuelva como parámetro de salida. Esta función deberá leer por teclado: el número de elementos diferentes de la fórmula (entre 1 y 5), así como los símbolos de los diferentes elementos (estos deberán estar en la tabla periódica) y su número de átomos (mayor que cero y menor de 100).
4. Una función que imprima en pantalla una ecuación química dada como argumento, con el mismo formato de los ejemplos presentados anteriormente.
5. Una función para validar ecuaciones químicas que acepte como parámetro de entrada una ecuación química, y que devuelva a través de su identificador un código entero con valor 1 (ecuación correcta) o 0 (ecuación no válida). Dicha función tan solo deberá comprobar si coinciden o no los números de átomos de cada elemento atómico a ambos lados de la ecuación.