Universidad de Alcalá

Departamento de Automática

INFORMÁTICA

Práctica 4.

Programación básica en C.

Grado en Ingeniería en Electrónica y Automática Industrial

Curso 2016-2017

v2.2 (23.09.2016)

A continuación figuran una serie de ejercicios propuestos, agrupados por dificultad. La práctica consiste en hacer la mayor cantidad de ellos posible, teniendo en cuenta que los que se propongan para el examen de laboratorio serán muy parecidos. Para cada uno de ellos, se pide cumplir las especificaciones completamente. Para aquellos detalles del ejercicio que no se especifiquen en el enunciado se admite la decisión justificada del alumno.

Para todos los ejercicios, y salvo que se especifique en el propio enunciado, utilizar la entrada estándar para obtener los datos, y la salida estándar para mostrar los resultados.

Bloque 1.

1. Calcular la suma de dos matrices. Las matrices tendrán unas dimensiones máximas especificadas por constantes. El programa solicitará al usuario las dimensiones de las matrices, y comprobará que no superan los límites especificados por las constantes. La matriz resultado se mostrará fila por fila.
2. Clasificar los números de una lista en pares e impares. El resultado del programa serán dos listas, una con los pares y otra con los impares. Las listas tendrán un tamaño máximo especificado por medio de constantes. El programa debe asegurarse de que el número de datos leídos no supera el espacio disponible.
3. Escribir la sucesión de Fibonacci. El programa debe leer dos números enteros de teclado. El primer dato leído será el número de elementos de la secuencia a mostrar. El segundo, el elemento de la sucesión a partir del que hay que mostrar, es decir, a partir de ese elemento, el programa deberá mostrar tantos elementos de la sucesión como indique el primer número leído. La sucesión de Fibonacci es:
    1. $n_0=0$;
    2. $n_1=1$;
    i. $n_i=n_{i-1}+n_{i-2}$;
4. Invertir el orden de los caracteres de una cadena de texto. Se mostrarán la cadena original y la cadena invertida. Las cadenas tendrán una longitud máxima especificada por una constante. El programa debe asegurarse de que los caracteres de la cadena no superan el espacio disponible.

Bloque 2.

1. Visualizar un entero en binario codificado en ASCII (*). Se visualizarán tanto el entero leído como el resultado de su conversión.
2. Tomando como entrada una cadena de texto, extraer las consonantes. Mostrar ambas cadenas. Las cadenas tendrán una longitud máxima especificada por una constante. El programa debe asegurarse de que los caracteres de la cadena no superan el espacio disponible.
3. Convertir una matriz dispersa a formato comprimido. Mostrar ambas matrices. La matriz original se mostrará fila por fila. La matriz comprimida es una única lista de tripletas (valor, fila, columna). El formato comprimido consiste en almacenar en una única lista únicamente aquellos elementos de la matriz distintos de cero, conjuntamente con sus índices de fila y columna. Ejemplo (recordar que en C, los índices empiezan en 0):

matriz original:

$$\begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 4 \\ 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 \end{bmatrix}$$

matriz comprimida:

(1,0,1),(4,1,3),(2,3,0)

Las dimensiones máximas de la matriz y de la lista de tripletas se fijarán por medio de constantes. El programa debe asegurarse de que el número de elementos no supera el espacio disponible.

4. De una matriz, extraer su diagonal principal. Mostrar la matriz original, fila por fila, y la diagonal en una única lista. Las dimensiones máximas de la matriz se fijarán por medio de constantes. El programa debe asegurarse de que el número de elementos no supera el espacio disponible.

5. De una matriz, extraer su diagonal secundaria. Mostrar la matriz original, fila por fila, y la diagonal secundaria en una única lista. Las dimensiones máximas de la matriz se fijarán por medio de constantes. El programa debe asegurarse de que el número de elementos no supera el espacio disponible.
6. Dados una matriz y un vector, calcular el vector que se obtiene al multiplicar la matriz por el vector. Mostrar únicamente el vector resultado. Las dimensiones máximas de la matriz y del vector se fijarán por medio de constantes. El programa debe asegurarse de que el número de elementos no supera el espacio disponible.
7. Mostrar una lista con los números primos que se encuentran entre dos números dados.
8. Hacer un histograma de las vocales de un texto. Se leerá un texto, y se mostrará el número de veces que aparece cada vocal en él. Opcionalmente, el texto se puede almacenar en una cadena. En ese caso, la cadena tendrá una longitud máxima determinada por una constante. El programa deberá asegurarse de que el número de caracteres leídos no excede el espacio disponible.
9. Contar el número de palabras de un texto. Se leerá un texto y se mostrará cuántas palabras tiene. Considerar que el texto contiene únicamente letras, espacios en blanco, y signos de puntuación. Son signos de puntuación válidos la coma, el punto y coma, el punto y los dos puntos. Tras un signo de puntuación podrá o no haber un espacio en blanco. Opcionalmente, el texto se puede almacenar en una cadena. En ese caso, la cadena tendrá una longitud máxima determinada por una constante. El programa deberá asegurarse de que el número de caracteres leídos no excede el espacio disponible.

10. Dibujar en pantalla una pirámide construida con el carácter '*'. El programa leerá, en primer lugar, los pisos que tiene la pirámide, y la mostrará centrada sobre su base. Ejemplo:

```
4 pisos:
   *
  ***
 *****
*******
```

11. Llevar un histograma de resultados del evento "tirar un dado". Un histograma es una lista que cuenta, para cada elemento, el número de veces que ha salido ese resultado. Por ejemplo, la lista (el histograma): `5,3,7,2,6,1` indicaría que el `1` ha salido 5 veces, el `2` 3, el `3` 7 y así sucesivamente. Para ello, utilizar la función `random` y asociadas, presente en la biblioteca `stdlib.h` (consultar el manual).
12. Repetir el ejercicio 11, pero simular la tirada de dos dados.

Bloque 3.

1. Realizar la multiplicación de dos matrices. Se leerán dos matrices y se mostrará el resultado. Se debe comprobar que la multiplicación es posible. Las dimensiones máximas de las matrices se fijarán por medio de constantes. El programa debe asegurarse de que el número de elementos no supera el espacio disponible.
2. Contar el número de veces que un determinado patrón aparece en un texto. Se leerá en primer lugar un patrón, y a continuación el texto, y se mostrará el número de veces que aparece. Opcionalmente, el texto y el patrón se pueden almacenar en sendass cadenas. En ese caso, las cadenas tendrán una longitud máxima determinada por sendas constantes. El programa deberá asegurarse de que el número de caracteres leídos no excede el espacio disponible.
3. Ordenar una lista de palabras. Se leerán las palabras de la lista, y a continuación se mostrará la lista ordenada alfabéticamente. Opcionalmente, el texto se puede almacenar en una cadena. En ese caso, la cadena tendrá una longitud máxima determinada por una constante. El programa deberá asegurarse de que el número de caracteres leídos no excede el espacio disponible.
4. Implementar una calculadora básica (suma, resta, multiplicación y división) que permita trabajar en binario, octal, hexadecimal y decimal. Se permitirá al usuario especificar el sistema de representación previamente a la introducción de los datos. La selección de la operación a realizar se puede solicitar por menú. El resultado se mostrará en el mismo sistema de representación en el que se introdujeron los datos. Es preciso evitar las divisiones por cero. El programa repite continuamente su funcionamiento hasta que el usuario pulse una tecla determinada. Para este ejercicio, utilizar exclusivamente el teclado y la pantalla como entrada y salida estándar, respectivamente.

5. Contar el número de bits a 1/0 en un número de tipo `long int`. Mostrar únicamente el número de bits a 1.
6. Generar el bit de paridad de un entero.
7. Dado un número entero, calcular el número que se obtiene de invertir el orden de sus bits. El programa leerá el número, y a continuación mostrará los bits del número original y los del número resultado, ambos en binario codificado en ASCII (*).
8. Sustituir un patrón por otro en una cadena. El programa leerá un texto, a continuación el patrón a sustituir, y por último el patrón por el que debe sustituirlo. Mostrará la cadena modificada. Si no se ha encontrado el patrón, se mostrará la cadena sin modificar. Opcionalmente, los textos se pueden almacenar en cadenas. En ese caso, cada cadena tendrá una longitud máxima determinada por una constante. El programa deberá asegurarse de que el número de caracteres leídos no excede el espacio disponible.
9. Escribir un filtro: el programa leerá un patrón, y a continuación un texto multilínea. Mostrará sólo aquellas líneas del texto que contengan el patrón. Opcionalmente, los textos se puede almacenar en cadenas. En ese caso, cada cadena tendrá una longitud máxima determinada por una constante. El programa deberá asegurarse de que el número de caracteres leídos no excede el espacio disponible.
10. Hacer un "traductor" básico. El programa cargará de un archivo una serie de correspondencias entre palabras. A continuación, para cada palabra leída mostrará su traducción (si existe; en caso contrario la dejará igual). El proceso se repetirá hasta que se lea una palabra clave predefinida.

Bloque 4.

1. Resolver una sopa de letras. El programa cargará de un archivo una matriz de caracteres. A continuación, leerá una palabra y la buscará en la matriz en cualquier orientación (vertical, horizontal, diagonal). Si la encuentra, indicará su situación por medio de las coordenadas de su letra de inicio y las de su letra de fin. Si no la encuentra, lo indicará convenientemente.
2. Hacer una calculadora básica. El programa leerá una cadena de texto con las operaciones a realizar en notación polaca inversa (por ejemplo: "3 5 + 8 *" corresponde a `(3+5)*8)`, y mostrará el resultado. Suponer que la sintaxis de la línea es correcta.
3. Gestionar una base de datos sencilla. Los elementos de la base de datos tendrán un campo código (entero), un campo nombre (cadena de caracteres), y un campo edad (entero). Las únicas operaciones admitidas son: insertar nuevo elemento, eliminar elemento, y mostrar elemento. Insertar crea un nuevo elemento, y solicita los campos del mismo. Eliminar solicita un código, y, si está presente en la lista, lo elimina. Mostrar solicita un código y muestra la información asociada al elemento correspondiente. Suponer que el código de cada elemento es único.

NOTAS:

(*) El binario codificado en ASCII es un código que representa cada bit del dato por un carácter ASCII, bien el ‘0’, bien el ‘1’, dependiendo del valor del bit. De esta manera, un número entero se representa por una cadena de caracteres ASCII. Por ejemplo, el número decimal 5, en binario puro de 8 bits, se representaría así:

```
00000101.
```

Por lo tanto, en binario codificado en ASCII, se utilizaría un código ASCII para cada bit del número:

```
'0' '0' '0' '0' '0' '1' '0' '1' '0',
```

donde `'0'` representa el código ASCII del `0` y `'1'` el del `1`.