EJERCICIOS TEMA 8

1. La memoria

1) Disponemos de un computador con una memoria principal de 16 Megabytes. Indica cuántos bits son necesarios para el bus de direcciones y el bus de datos en los siguientes casos:

1. La memoria se lee y se escribe en palabras de 8 bits.
2. La memoria se lee y se escribe en palabras de 16 bits.
3. La memoria se lee y se escribe en palabras de 32 bits.

Resolver el problema planteado considerando dos supuestos: 1) Sólo disponemos de módulos de memoria cuyo tamaño de palabra es de un byte (el mínimo número de bloques), permitiéndose el acceso o direccionamiento a cada byte, y 2) en cada caso planteado la memoria (un único bloque) tiene la misma anchura que las palabras que se leen y se escriben.

2) Sea un vector en C que almacena números enteros de 32 bits cada uno. El vector contiene 300 elementos y está almacenado a partir de la posición 0x000B A89F de memoria. ¿A partir de qué dirección está almacenado el elemento del vector con el índice 100? ¿En qué dirección de memoria comenzará el último elemento de la tabla? Supóngase que cada fila de memoria es de un byte.

2. Aritmética de punteros

3) ¿Cuántas posiciones de memoria se reservan con la instrucción de C *vector = (int *) malloc(sizeof(short int)*100)*? Compruébalo con el compilador y justifica la respuesta.

4) Realizar un programa en C para calcular el tamaño que ocupa una variable de cada uno de los Tipos Básicos en C, así como el tamaño que ocupan las variables de tipos compuestos, como por ejemplo las variables de tipo *short int*. Para ello, utiliza la función *sizeof()*.

5) Dada la siguiente secuencia de sentencias en C:

```
#include <stdio.h>

int main (void){

  int x,*p;

  x=5;
  p=x;

  return 0;
}
```

¿Contiene *p* alguna dirección definida? Si es así, ¿cuál? ¿Qué dato estaría contenido en dicha dirección? Comprueba las respuestas mediante un programa que imprima el valor de x, p y *p antes de las asignaciones y después de las asignaciones.

6) Dada la siguiente secuencia de sentencias en C:

```
#include <stdio.h>

int main (void){

  int x,y,*p1,*p2;

  x=5;
  p1=&x;
  p2=*&p1;
  y=*&x;

  return 0;
}
```

¿Alguna de estas sentencias es errónea? Si no es así, ¿qué efecto producen las dos últimas asignaciones? Comprueba las respuestas mediante un programa que imprima el valor de x, y, &x, &y, p1, p2, *p1 y *p2 antes y después de las dos últimas asignaciones.

7) Realizar una función en C que dada una cadena de caracteres pasada como parámetro a una función, localice la primera aparición de un carácter en concreto y devuelva, al programa principal, el carácter siguiente a éste, así como la posición de este último, empleando aritmética y notación de punteros. **Nota**: la función debe devolver dos valores, uno de tipo

char y otro de tipo *int*. Realizar un programa principal para comprobar el correcto funcionamiento de la función realizada.

8) Realizar un programa en C que permita obtener los bytes de un número real (tipo *float* en C, almacenado en formato IEEE 754). Imprime el signo, la característica y la fracción de dicho número. Modifica dicho programa para que, dado el signo, la característica y la fracción, introducidos por el usuario, se almacene en la máquina de forma correcta el número real en formato IEEE 754. Comprueba que sea correcto imprimiendo posteriormente el valor del número.

3. Creación y gestión de vectores y matrices con memoria dinámica

9) Sea m una matriz de 10x10 enteros. Indica qué representan cada una de las siguientes notaciones:

- m[0][9]
- m[1]
- &m[2][0]
- m+1
- m[3]+1

10) Realizar un programa en C que reserve memoria para un vector de enteros cuyo tamaño es solicitado al usuario del programa, lo rellene, y calcule el máximo y el mínimo de los elementos almacenados en él. Todas estas operaciones se realizarán mediante funciones, es decir, tendremos una función para reservar la memoria, otra para rellenar el vector, otra para calcular el valor máximo y finalmente otra para calcular el valor mínimo. La liberación de la memoria reservada también se realizará mediante una función.

11) Realizar un programa en C que, mediante un menú mostrado por pantalla, realice las siguientes operaciones: rellenar un vector de caracteres de tamaño 8 creado mediante memoria dinámica, imprimirlo por pantalla, modificar una celda del mismo y obtener el valor de una posición en concreto. Todas estas operaciones se deberán realizar empleando funciones. La gestión de la memoria deberá realizarse de forma correcta.

12) Realizar un programa en C que declare una matriz de tamaño 3xn, donde n será un valor leído por teclado y solicitado al usuario en tiempo de ejecución, la rellene de manera

automática (ver función *rand()* en el enunciado de la práctica 4), y calcule el valor máximo de la matriz y el valor mínimo de una columna solicitada al usuario del programa. Todas estas operaciones se realizarán mediante funciones, y se realizará una gestión óptima de la memoria.

13) Implementar una función en C que tome dos cadenas de caracteres como entrada, y devuelva las cadenas mezcladas de forma inversa. Por ejemplo, si las cadenas son c1 = "HOLA" y c2 = "informatica" devolvería la cadena "AaLcOiHtamrofni". Hacer un programa en C para probarla. Deberá emplearse memoria dinámica para resolver el problema, tanto para el almacenamiento de las cadenas c1 y c2, cuyo contenido será introducido por el usuario en tiempo de ejecución, como la cadena resultante de la unión de ambas; que tendrá que tener el tamaño exacto para evitar consumir más memoria de la necesaria. La gestión de la memoria deberá ser óptima.

14) Realizar una función en C que dada una cadena de caracteres introducida por teclado corte la misma y genere una subcadena a partir de ésta desde la posición i a la posición j, ambas incluidas. La función devolverá, mediante memoria dinámica, la subcadena generada. Realizar un programa principal en C que compruebe el funcionamiento de la función realizada. Como mínimo, el programa principal solicitará al usuario la cadena a cortar y los índices, y deberá mostrar por pantalla la subcadena resultado.

15) Implementar una función que sea equivalente a la función de C *int strlen (char *)*. Comprobar su funcionamiento mediante un programa principal. Deberá resolverse mediante memoria dinámica.

16) Escribir un programa en C que declare una matriz entera 5x5 de forma estática y de forma dinámica. Obtén las direcciones de memoria en las que se almacenan los elementos de la misma y realiza una representación gráfica de las direcciones obtenidas.

17) Escribir un programa en C que, empleando memoria dinámica, permita leer una frase de tamaño máximo N, solicitada al usuario en tiempo de ejecución. Posteriormente, deberá mostrarla con un carácter en cada línea.

18) Escribir un programa en C que, empleando memoria dinámica, permita leer una frase de tamaño máximo N y cuente el número de vocales de cada tipo que aparecen en ella. El resultado deberá mostrarse por pantalla.

19) Escribir un programa en C que, empleando memoria dinámica y dada una frase de tamaño máximo N, determine si es un palíndromo o no. Un palíndromo es una frase que, atendiendo sólo a sus letras e ignorando los espacios, acentos, signos de puntuación y tipo de letra (mayúscula o minúscula) expresa lo mismo leída de izquierda a derecha que de derecha a izquierda. Ejemplo: “dábale arroz a la zorra el abad”.

20) Realizar un programa en C que solicite al usuario el número de alumnos de que consta una clase. Posteriormente, se creará un vector que contendrá las edades de los alumnos de una clase, que serán inicializadas empleando la función *rand()*. Las edades deberán estar comprendidas entre los valores 18 y 28. Finalmente, calcular la edad media y el alumno de menor edad mediante funciones que devuelvan al programa o función principal los resultados, que serán mostrados por pantalla.

21) Un psiquiatra tiene un paciente que habla invirtiendo las frases completas. Por ejemplo: si quiere decir “hola caracola”, lo que realmente dice es: “alocarac aloh”. Para poder comunicarse con el paciente se decide hacer un programa que traduzca lo que él dice al lenguaje del paciente. Implementar dicho programa en C. Hacer dos versiones: la primera empleando vector auxiliar, en la segunda la inversión deberá realizarse en el mismo vector. Con memoria dinámica declararemos cuál será la longitud máxima de la frase que podrá ser leída.

22) Escribir un programa en C que lea el nombre y los dos apellidos de una persona (en tres cadenas de caracteres diferentes), y posteriormente deberán unirse en una única cadena declarada como memoria dinámica, añadiendo un espacio entre el nombre y el primer apellido, y entre éste y el segundo apellido.

23) Escribir un programa en C que cuente el número de palabras en un texto de tamaño máximo N, siendo N un valor introducido por el usuario.

24) Escribir una función en C que reciba como parámetros de entrada un vector de n números enteros y dos posiciones i y j (i<=j) que determinen un subvector de j-i+1 elementos, y devuelva la posición de ese subvector en la que se encuentre el entero más pequeño. El vector debe ser creado en tiempo de ejecución una vez el usuario ha determinado el valor de n.

25) Escribir una función en C que ordene los elementos de un vector de n números enteros que se le pasa como parámetro, a la vez de entrada y salida. No se debe usar un vector auxiliar y el vector debe ser creado en tiempo de ejecución una vez que el usuario ha determinado el valor de n.

26) Escribir un programa que cree dinámicamente tres matrices A, B y C de números naturales cuyos tamaños deben ser proporcionados por el usuario al inicio de la ejecución. Se rellenarán las matrices A y B con valores aleatorios entre 0 y 4, y se calculará la multiplicación de A y B almacenando el resultado en la matriz C.