Universidad Rey Juan Carlos **Curso 2005/06**

Ingenierías Técnicas de Informática de Sistemas y de Gestión

Asignatura: Programación en C/C++

Hoja de Problemas Nº 3: Funciones, ficheros y otros aspectos de C.

Normas generales

- La entrega de esta hoja de ejercicios es *voluntaria* y debe realizarse *individualmente*.
- La fecha límite de entrega de la práctica será el martes, día *18 de abril de 2006*, en el horario de clase.
- Hay que entregar:
  a) Un disquete, *etiquetado con los apellidos y nombre del alumno/a*, conteniendo los ficheros *fuente* en C de las soluciones a los ejercicios propuestos
  b) Las soluciones a los ejercicios en papel.

Responda a las siguientes preguntas:

1) Explicar clara y brevemente qué hace el siguiente programa en *C*. Mostrar el resultado producido mediante un ejemplo:

```
#include <stdio.h>
#define fil 5
#define col 5
void misterio1(int(*)[col], int, int);
int *misterio2 (int(*)[col], int, int);

void main()
{
  int p, q, array[fil][col];
  int *pfila, i;
  do {
     printf ("Dame dato: ");
     scanf ("%d",&p);
  } while (p<2 || p>5);
  do {
     printf ("Dame dato: ");
     scanf ("%d", &q);
  } while (q<2 || q>5);
  misterio1 (array, p, q);
  pfila = misterio2 (array, p, q);
  for (i=0; i<q; i++)
    printf(" %d ", *(pfila+i));
}
```

```
void misterio1 (int (*array)[col], int p, int q)
{
 int i, j;
 for (i=0; i<p; i++)
   for (j=0; j<q; j++)
   {
    printf ("Dame valor [%d,%d]: ", i, j);
    scanf ("%d", (*(array+i)+j));
   }
}

int *misterio2 (int (*array)[col], int p, int q)
{
 int i, j, suma=0, max, pos;
 for (i=0; i<p; i++)
 {
   for (j=0; j<q; j++)
     suma += *(*(array+i)+j);
   if (!i)
    {
       max = suma;
       pos = i;
    }
   else
       if (suma > max)
        {
           max = suma;
           pos = i;
        }
      suma = 0;
    }
 return (*(array+pos));
}
```

2) Explicar clara y brevemente qué hace el siguiente programa en *C*. Mostrar el resultado producido mediante un ejemplo:

```
#include <stdio.h>
void misterio (int [ ][10], int, int);

void main()
{
   int m[10][10], rango, a;
   do {
        printf ("¨Rango (1-10)?");
        scanf ("%d", &rango);
   } while (rango<=0 || rango>10);

   printf ("Valor de a:");
   scanf ("%d", &a);
   misterio (m, rango, a);
}
```

```
void misterio (int t [ ][10], int r, int n)
{
  int tope, a, i, k;
  if (r%2)
     tope =r/2+1;
  else
     tope=r/2;
  for(a=0;a<tope;a++)
  {
     for(i=a,k=a; k<r-a; k++, n++) t[i][k]=n;
     for(i=a+1,k--; i<r-a; i++,n++) t[i][k]=n;
     for(k--,i--; k>=a; k--,n++) t[i][k]=n;
     for(k++,i--; i>a; i--,n++) t[i][k]=n;
  }
}
```

Desarrolle programas en *C* para los siguientes problemas propuestos:

3) Escritura de un subprograma *Sustituir* que recibe un puntero a una cadena de caracteres como argumento, y reemplaza en la cadena todos los espacios en blanco ‘ ‘ por un carácter ‘-‘, devolviendo también dicho subprograma el número total de caracteres sustituidos. Por ejemplo, si se recibe la cadena:

   “hoy es el primer jueves de abril”

   se tiene que devolver la cadena:

   “hoy-es-el-primer-jueves-de-abril” y el valor 6 (número de sustituciones).

4) Codificación de una función booleana que reciba una cadena de caracteres representando una palabra y decida si ésta es *palíndroma*, es decir, si se lee igual de izquierda a derecha que de derecha a izquierda. Por ejemplo, la palabra “salas” es palíndroma.

5) Sea un texto de longitud variable (hasta un máximo de 50 caracteres), formado solo por caracteres imprimibles. Dicho texto se lee por pantalla y se almacena en una lista enlazada donde cada carácter se guarda en un nodo de la lista en el mismo orden en el que aparece en el texto. Se desea eliminar de dicha lista *exclusivamente* los caracteres blancos, es decir hay que devolver la misma lista de caracteres (en el mismo orden que originalmente) pero sin los nodos correspondientes a los blancos. Codificar un programa en C para resolver este problema. (Nota: Se presupone que el texto no puede empezar por un carácter blanco, aunque puede haber un número variable de caracteres blancos entre las palabras del texto)..

6) Escribir un programa en C para decidir si un carácter aparece en una cadena de caracteres.

7) Sea un polinomio simbólico en una variable, donde los coeficientes de sus términos son enteros y los exponentes de dichos términos son enteros positivos. Un ejemplo de estos polinomios puede ser el siguiente: $3x^{10} - 4x^2 + 11$.

   Se pide codificar en **C** lo siguiente:

   a) una representación *razonable* del *TipoPolinomio* descrito en el párrafo anterior, y
   b) una función que permita construir un polinomio a partir de los datos correspondientes,
   c) una función que realice la suma de dos polinomios culaesquiera, y
   d) el programa principal que llame a los demás subprogramas.

8) Escribir un programa en C para sumar dos matrices cuyo tamaño puede variar de una ejecución a otra (*matrices dinámicas*). El programa deberá leer las dimensiones de cada matriz, comprobar que las matrices leídas son de las mismas dimensiones, reservar memoria, leer cada una de las matrices, sumarlas y escribir el resultado. Finalmente, se liberará la memoria de cada matriz definida. Es importante desarrollar este programa de forma *modular*, es decir definiendo las funciones apropiadas para realizar cada subtarea (p. ej. *LeerMatriz, SumarMatrices*, etc.). El programa principal comprobará el funcionamiento correcto de las funciones definidas.

9) Desarrolle un programa que reciba dos ficheros secuenciales de números enteros ordenados de manera creciente y genere un nuevo fichero de salida con los números de ambos también ordenados crecientemente. Se deberán elegir los formatos apropiados para los datos de los ficheros (estos ficheros serán abiertos apropiadamente, y al final del proceso cerrados). El número de datos en cada fichero de entrada puede ser variable. En el fichero de salida se escribirán tres números por línea. Por ejemplo, para los ficheros de entrada *fent1* y *fent2*,

   fent1:
   ```
   3  5  8
   17  24  33
   ```

   fent2:
   ```
   1  5
   7  16
   20 22  34
   ```

   el fichero resultado *fsal* será:

   fsal:
   ```
   1  3  5
   5  7  8
   16 17 20
   22 24 33
   34
   ```

10) Escribir un programa en **C** que permita leer y procesar las *n* líneas de un fichero de texto según se describe a continuación. Para cada línea leída del fichero hay que: reservar memoria acorde con su longitud, guardar su contenido, contar las vocales de cada tipo que hay en dicha línea e imprimirla por pantalla junto con su número de orden en el texto (previo a la línea correspondiente) y, finalmente, en la siguiente línea se escribe la cuenta de las vocales realizada para la línea procesada actualmente. Hay que mantener en la salida por pantalla el formato del fichero de datos.

A modo de ejemplo, para el fichero de texto "prueba.txt" cuyo contenido es:

```
KRISHNAMURTHI, the Penguin Krishnamurthi Reader
If we can really understand the problem,
the answer will come out of it,
because the answer is not separate from the problem.
```

la salida por pantalla será *exactamente* de la forma:

```
1) KRISHNAMURTHI, the Penguin Krishnamurthi Reader
num. a = 3; num. e = 4; num. i = 5; num. o = 0; num. u = 3
2) If we can really understand the problem,
num. a = 3; num. e = 5; num. i = 1; num. o = 1; num. u = 1
3) the answer will come out of it,
num. a = 1; num. e = 3; num. i = 2; num. o = 3; num. u = 1
4) because the answer is not separate from the problem.
num. a = 4; num. e = 8; num. i = 1; num. o = 3; num. u = 1
```

11) Un fichero de texto contiene en cada línea dos cadenas de texto que representan números enteros, separadas por el operador (carácter) '+' o por el operador '-', existiendo también al menos un carácter blanco entre cada número y el operador correspondiente. Se desea construir un *fichero binario* que contenga los resultados de las operaciones que aparecen en cada línea del fichero, aplicados a los datos de la línea correspondiente.

12) Se desea implementar una estructura de datos *pila* en C utilizando memoria dinámica en C. Las estructura de datos tendrá asociada las siguientes operaciones: *CrearVacia*, *InsertarElemento* y *DevolverCima.* Desarrollar un programa modular en C para implementar esta estructura de datos y hacer uso de ella. El programa estará formado por dos ficheros que se compilarán por separado: uno contendrá la implementación de las operaciones indicadas y el programa principal llamará a dichas operaciones.