UNIVERSIDAD PONTIFICIA ICAI ICADE COMILLAS MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

Fundamentos de Informática 1ºIIND

Examen Junio 2006

Programa 1 (2 puntos)

Una matriz de caracteres guarda un conjunto de letras que podrían formar palabras. Se trata de escribir el **programa principal** y unas **funciones** que busquen palabras en dicha matriz (función `Diagonales` y función `Buscar`).

El programa principal debe realizar las siguientes tareas:

- Declarar una matriz estática de tamaño 100x100, que luego sólo estará parcialmente llena pero será cuadrada
- Preguntar al usuario con qué dimensión se quiere trabajar. Comprobad que la dimensión sea menor que 100 y mayor que 2, en caso contrario dar un mensaje de error y volver a preguntar.
- Preguntar al usuario las letras de la matriz
- Llamar a la función Diagonales que obtiene las cadenas de caracteres correspondientes a las dos diagonales:

  ```
  void Diagonales(char mat[][100], int fil, int col, char *diag1, char *diag2);
  ```
- Mostrar las palabras de las diagonales
- Preguntar al usuario una palabra de dos letras. Comprobad que realmente la longitud es 2 y en caso contrario dar un mensaje de error y volver a preguntar.
- Llamar a la función `Buscar` que busca en la matriz la palabra de dos letra, bien en dirección horizontal (sólo de izquierda a derecha) o bien en dirección vertical (sólo de arriba hacia abajo). La función devolverá 1 si ha encontrado la palabra y 0 en caso contrario.

  ```
  int Buscar(char mat[][100], int fil, int col, char *pal);
  ```
- Mostrar el mensaje “Encontrado” o “No Encontrado”.

Ejemplo:

h	s	a	p
t	o	e	f
k	p	l	y
e	m	j	a

```
Diagonal1: hola
Diagonal2: pepe
Palabra a buscar? so
Encontrado
```

Programa 2 (3 puntos)

Simulación de un terminal TPV (Terminal de Punto de Venta) – Creación de un ticket de compra en la tienda de deportes de un gran almacén.

Introducción

Los TPVs se encuentran situados en los mostradores de salida de supermercados, grandes superficies y tiendas en general. De una forma simple, desde el punto de vista de hardware podemos decir que un TPV es un PC al que se le ha añadido un lector de códigos de barras. Desde el punto de vista de software tiene los debidos programas de control y, lo que es más importante en nuestro caso, la información de los productos que se venden en la tienda en la que está instalado el TPV. El TPV es el que procesa con todo ello, HW+SW, el paso de los artículos comprados y produce también el ticket de compra.

Si pensamos en un gran almacén todos los artículos que comercializa estarán en un fichero binario **`articulos.dat.`** Por cada artículo hay un registro y la organización de cada registro se corresponde con la siguiente estructura:

```
typedef struct
{
        char descripcion [20];
        char codigo [12];
        int departamento;
        char nombre_proveedor[20];
        char nif_proveedor[10];
        double precio_unitario;
}T_ARTICULO;
```

De este fichero, para el TPV de la tienda de deportes, se extraen sólo los artículos deportivos (número de departamento **1111**) y se guardan en otro fichero binario **`deportivos.dat`** con una estructura más reducida, como se muestra a continuación:

```
typedef struct
{
        char descripcion_item [20];
        char codigo_item [12];
        double precio_unitario;
}T_DEPORTIVOS;
```

Se pide:

- **Programa principal**

Llamar a la función **`Crear_Fichero_Deportes()`** que crea el fichero binario **`deportivos.dat`** a partir del fichero **`articulos.dat`**, extrayendo la información de los registros de éste cuyo número de departamento sea **`1111`**.

Pedir —con el número de registros grabados en **`deportivos.dat`**— memoria dinámica para crear un vector de estructuras y cargar la información del fichero **`deportivos.dat`**.

Llamar a la función **`Crear_Ticket ()`** que con la información del vector de estructuras que se le pasa creará el ticket de compra. Está función se ejecutará un número indeterminado de veces hasta que el usuario decida terminar.

- **Funciones**

Crear_Fichero_Deportes ()

Crea el fichero **`deportivos.dat`** a partir del fichero **`articulos.dat`** y devuelve el número de registros creados en el primer fichero. Hay que prever que no haya ningún registro de artículos deportivos en **`articulos.dat`** y la circunstancia debe indicarse al usuario.

Crear_Ticket ()

Con la información contenida en el vector de estructuras de artículos deportivos produce el ticket de compra.

Se simulará el proceso introduciendo por teclado el código del artículo (imitación de la lectura del código de barras) y el número de artículos adquiridos. El programa verificará que el código está en el vector de estructuras y se emitirá un mensaje de error por pantalla en el caso de que no esté. En caso afirmativo el programa escribirá un registro en el fichero de texto **`ticket.txt`**, un registro por línea que incluye la descripción del artículo, código, número de unidades vendidas y el precio total. Se mantendrá un total actualizado con la intención de producir un total de la compra como última línea del ticket.

El proceso descrito en el párrafo anterior continuará hasta que el paso de los artículos comprados a través del TPV se acabe al introducir un código de artículo ficticio **`XXX-XXX-XXX`**. En ese momento, basándose en la información existente en el fichero de texto, se imprimirá el ticket de compra por pantalla que constará de una cabecera y después una línea por articulo, finalizando con una línea final con el importe total de la compra.

Ejemplo de ticket de compra:

```
           BLUMENTHAL DEPARTMENT STORE
         ---------------------------------

Descripción           Código        Cant.  Cargo
--------------------- ------------ -----  ---------
Botas esquí           fff-222-333    1       380.70
Polainas              zzz-333-444    3       110.70
Calcet. send.         bbb-222-333    4        72.00
Calcet. maratón       vvv-999-000    2        37.40


                                   TOTAL       600.80

                        ¡¡GRACIAS POR SU VISITA!
```

```c
/* Solución del programa 1.  Ex JUN 2006
   Rafael Palacios
*/
#include <stdio.h>
#include <string.h> //para strlen

void Diagonales(char mat[][100], int fil, int col, char *diag1, char
*diag2);
int Buscar(char mat[][100], int fil, int col, char *pal);

int main(void)
{
    char mat[100][100]; /* matriz dato */
    int row,col; /* número de filas y columnas */
    char diag1[100], diag2[100]; /* diagonales */
    char palabra [100];  /* leyendo con fgets basta con tamaño 3 */
    int encontrado;  /* resultado de la función Buscar */
    int i,j;

    /* Preguntar dimensión de la matriz */
    do  {
        printf("Dimensión de la matriz: ");
        scanf("%d",&row);
        if (row>=100 || row<=2)  {
            printf("Error la dimensión no es válida\n");
        }
    } while (row>=100 || row<=2);
    col=row;  /* el enunciado dice matriz cuadrada, no pregunto col */

    /* Preguntar las letras de la matriz */
    for(i=0; i<row; i++)  {
        for(j=0; j<col; j++)  {
            printf("Elemento %d,%d\n",i+1,j+1);
            scanf(" %c",&mat[i][j]);
        }
    }

    /* Calculo diagonales */
    Diagonales(mat,row,col,diag1,diag2);

    /* Muestro resultados */
    printf("Diagonal 1: %s\n",diag1);
    printf("Diagonal 2: %s\n",diag2);

    /* Pregunto una palabra de dos letras */
    fflush(stdin); /* limpio buffer de techado */
    do  {
        printf("Palabra a buscar? ");
        fgets(palabra,4,stdin);  /* leo 4 máximo. Ejemplo: 's' 'o' '\n'
        '\0' */
        if (strlen(palabra)!=3 || palabra[2]!='\n')  {
            printf("Error, la palabra debe tener exáctamente 2 letras\n");
        }
    } while (strlen(palabra)!=3 || palabra[2]!='\n');
    palabra[2]='\0'; /* quito el \n final que pone fgets */

    /* Alternativa
    do  {
        printf("Palabra a buscar? ");
        scanf("%s", palabra);
        if (strlen(palabra)!=2) {
            printf("Error, la palabra debe tener exáctamente 2 letras\n");
        }
    } while (strlen(palabra)!=2);
    */

    /* Busco la palabra en la matriz */
    encontrado=Buscar(mat,row,col,palabra);

    /* Mostrar resultado de la búsqueda */
    if (encontrado==1) printf("Encontrado\n");
    else printf("No encontrado\n");
```

```
    return 0;
}


void Diagonales(char mat[][100], int fil, int col, char *diag1, char
*diag2)
/* contruye las cadenas diag1 y diag2 con las diagonales de la matriz mat
Entrada:
    mat: matriz con letras
    fil, col: dimensiones efectivas de mat
    diag1: diagonal principal (a construir)
    diag2: segunda diagonal (a construir)
Salida:
    Nada
Advertencias:
    Esta función es válida incluso si la matriz no es cuadrada
*/
{
    int i; /* contador */
    int dim; /* dimensión */

    if (fil<col) dim=fil;
    else dim=col;

    for(i=0; i<dim; i++)  {
        diag1[i]=mat[i][i];
    }
    diag1[i]='\0';  /* No olvidar el \0 de la cadena */

    for(i=0; i<dim; i++)  {
        diag2[i]=mat[i][col-1-i];
    }
    diag1[i]='\0';  /* No olvidar el \0 de la cadena */
}


int Buscar(char mat[][100], int fil, int col, char *pal)
/* busca una palabra en horizontal o vertal dentro de la matriz mat
Entrada:
    mat: matriz con letras
    fil, col: dimensiones efectivas de mat
    pal: palabra de dos letras
Salida:
    devuelve 1 si encuentra la palabra, 0 si no la encuentra
Advertencias:
    La palabra pal debe tener dos letras
    Esta función es válida para matrices cuadradas o rectangulares
*/
{
    int res; /* resultado de la función */
    int r,c; /* contadores */

    res=0;
    /* Búsqueda horizontal */
    for(r=0; r<fil; r++)  {
        for(c=0; c<col-1; c++)  {
            if (pal[0]==mat[r][c] && pal[1]==mat[r][c+1]) res=1;
        }
    }

    /* Búsqueda vertical */
    for(r=0; r<fil-1; r++)  {
        for(c=0; c<col; c++)  {
            if (pal[0]==mat[r][c] && pal[1]==mat[r+1][c]) res=1;
        }
    }

    return res;
}
```

```c
// Examen Final Teórico Junio 2006
// Curso 2005-2006
// Problema_2.c

// Preprocessor directives
#include <stdio.h>
#include <stdlib.h>
#include <ctype.h>
#include <string.h>
// Declaring structures
// Stucture corresponding to items.dat
typedef struct
{
     char description [20];
     char code [12];
     int department;
     char supplier name[20];
     char supplier id[10];
     double unit_price;
}T ITEMS;
// Structure corresponding to sports.dat
typedef struct
{
     char description [20];
     char code [12];
     double unit_price;
}T SPORTS;
// Prototypes of functions
int Create Sports File( );
void Create Ticket (T SPORTS *sport, int no articles);
int Check Code (T_SPORTS *sport, int no_articles, char *code);
void Clean String (char *string);
void Display_Sports_File();
void Display Sports Array (T SPORTS *sport, int no articles);
/**************************************************************************
*****/
int main (void)
{
     T SPORTS *sport;
     int no articles;           // Number of records of sporting goods in
     items.dat
     char new ticket;
     FILE *p sports;
     no_articles = Create_Sports_File( );
     Display Sports File();
     printf("\nArticles extracted from items.dat => %d\n", no_articles);
     if(no_articles<0)
     {
          printf("\nError in original file.\n");
     }
     else if (no_articles==0)
     {
          printf("\nNo sporting goods in the basic file.\n");
     }
     else
     {
          // Requesting dynamic memory for no_articles
          // Creating sports array
          sport=(T_SPORTS*) calloc (no_articles, sizeof(T_SPORTS));
          if(sport == NULL)
          {
               printf("\nERROR when allocating dynamic memory.\n");
          }
          else
          {
               // Open sports.dat for reading
               p sports = fopen("sports.dat", "rb");
               if(p_sports==NULL)
               {
                    printf("\nERROR.- Cannot open both designated
                    files.\n");
               }
```

```
                else
                {
                     // Load the entire file contents to an array of
                     structures
                     fread(sport, sizeof(T_SPORTS), no_articles, p_sports);
                     // Close sports.dat
                     fclose(p sports);
                     Display Sports Array (sport, no_articles);
                     // Create the  shopping ticket
                     do{
                          Create_Ticket(sport, no_articles);
                          printf("\nAnother ticket? Y/N ");
                          scanf(" %c", &new ticket);
                     }while (toupper(new_ticket)!='N');
                }
                free(sport);
           }
      }
      return 0;
}
/*************************************************************************
**/
int Create_Sports_File( )
{
      FILE *p items;
      FILE *p_sports;
      T ITEMS aux1;
      T SPORTS aux2;
      int no sports;
      no sports = -1;
      // Open items.dat for reading
      // Open sports.dat for writing and reading
      p_items = fopen("items.dat", "rb");
      p sports = fopen("sports.dat", "wb");
      if((p_items==NULL)||(p_sports==NULL))
      {
          printf("\nERROR.- Cannot open both designated files.\n");
      }
      else
      {
           // Creating the file sports.dat from the items.dat file
           no sports = 0;
           fread(&aux1, sizeof(T ITEMS), 1, p_items);
           while (feof(p_items)==0)
           {
                if(aux1.department==1111)
                {
                     no_sports++;
                     strcpy(aux2.description, aux1.description);
                     strcpy(aux2.code, aux1.code);
                     aux2.unit price = aux1.unit price;
                     fwrite(&aux2, sizeof(T_SPORTS), 1, p_sports);
                }
                fread(&aux1, sizeof(T_ITEMS), 1, p_items);
           }
           // Close items.dat file
           fclose(p items);
           fclose(p_sports);
      }
      return no_sports;
}
/*************************************************************************
*********/
void Create_Ticket (T_SPORTS *sport, int no_articles)
{
      char code [12];
      char description[20];
      int units;
      int result;
      FILE *p_text;
      int no_lines;
      double total;
```

```
     double grand_total;
     int i;
     int go;
     no_lines = 0;
     grand_total = 0.;

     // Open for both writing and reading
     p_text = fopen ("ticket.txt", "w+");
     if(p_text==NULL)
     {
          printf("\nError - Cannot open designated file.\n");
     }
     // Simulate a TPV transaction
     else
     {
          go = 1;
          do{
               fflush(stdin);
               printf("\nEnter code of article => ");
               fgets(code, 12, stdin);
               Clean String(code);
               if(strcmp(code,"XXX-XXX-XXX"))
               {
                    do{
                         printf("Enter units of item being bought => ");
                         scanf("%d", &units);
                    }while(units<=0);

                    result = Check_Code (sport, no_articles, code);
                    if(result<0)
                    {
                         printf("\nCode not found.\n");
                    }
                    else
                    {
                         fprintf(p_text, "%-20s %s %d %.2f\n",
                             sport[result].description, sport[result].code,
                             units,
                             units*sport[result].unit_price);
                         no_lines++;
                         grand_total+=units*sport[result].unit_price;
                    }
               }
               else
               {
                    go = 0;
               }
          }while(go==1);

          // printf ticket
          rewind(p_text);
          // system("cls");
          printf("\n\t   BLUMENTHAL DEPARTMENT STORE\n");
          printf("\t---------------------------------\n");
          printf("Description          Code         Units  Charge\n");
          printf("-------------------- ------------ -----  ---------\n");
          for(i=0; i<no_lines; i++)
          {
               fgets(description, 20, p_text);
               fscanf(p_text,"%s", code);
               fscanf(p_text,"%d", &units);
               fscanf(p_text,"%lf\n", &total);
               printf("%-20s %-12s %5d  %9.2f\n", description, code, units,
               total);
          }
          fclose(p_text);
          printf("\n\n                                    TOTAL  %10.2f",
          grand_total);
          printf("\n\n\t\t\tTHANK YOU FOR YOUR VISIT!!!");
     }
}
/*****************************************************************************
```

```
***/
int Check_Code (T_SPORTS *sport, int no_articles, char *code)
{
     int i;
     int found;
     found = -1;
     for (i=0; i<no_articles&&found==-1; i++)
     {
          if(strcmp(sport[i].code, code)==0)
               found=i;
     }
     return found;
}
/*************************************************************************
****/
void Clean_String (char *string)
{
     int i;
     i=0;
     while((string[i]!='\0')&&(string[i]!='\n'))
        i++;

     if(string[i]=='\n')
          string[i]='\0';
}
/*************************************************************************/
void Display_Sports_File()
{
     FILE *p sports;
     T SPORTS aux;
     p_sports = fopen("sports.dat", "rb");

     if(p_sports==NULL)
     {
          printf("\nERROR.- Cannot open read file sports.dat.\n");
     }
     else
     {
          printf("\n\t\t\t   CONTENTS OF SPORTS.DAT FILE\n");
          printf("\t\t\t-------------------------------\n\n");
          printf("Description          Code         Unit Price\n");
          printf("-------------------- ------------ ----------\n");

          fread(&aux, sizeof(T_SPORTS), 1, p_sports);
          while(feof(p_sports)==0)
          {
               printf("%-20s %-12s %10.2f\n", aux.description, aux.code,
                   aux.unit_price);
               fread(&aux, sizeof(T_SPORTS), 1, p_sports);
          }
          fclose(p_sports);
     }
}
/*************************************************************************
******/
void Display_Sports_Array (T_SPORTS *sport, int no_articles)
{
     int i;
     printf("\n\t\t\t   CONTENTS OF SPORTS ARRAY\n");
     printf("\t\t\t-------------------------------\n\n");
     printf("Description          Code         Unit Price\n");
     printf("-------------------- ------------ ----------\n");

     for (i=0; i<no_articles; i++)
     {
          printf("%-20s %-12s %10.2f\n", sport[i].description, sport[i].
          code,
               sport[i].unit_price);

     }
}
/*************************************************************************
```

```
******/
```