UNIVERSIDAD PONTIFICIA ICAI ICADE COMILLAS MADRID
ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

Fundamentos de Informática 1ºIIND
Examen Junio 2006

Nombre	**Apellidos:**
Grupo: □ **A** □ **B** □ **C** □ **D** □ **E**	

Advertencias:

1. Duración del examen 2 horas y 45 minutos
2. No se puede desgrapar el cuadernillo del examen.
3. Se puede utilizar lápiz.
4. Las funciones sólo pueden tener un `return`
5. Las funciones no pueden realizar operaciones de entrada/salida (`printf, scanf, gets`,…) salvo que se trate de funciones específicas para leer datos o para mostrar resultados.
6. No se puede utilizar `exit` (salvo errores excepcionales como memoria dinámica y archivos), ni `continue`, ni `break` (salvo en `switch`)
7. No se pueden utilizar variables globales

Calificación:

Preguntas Test (3 puntos)	**Códigos cortos (2 puntos)**	**Programa 1 (2 puntos)**	**Programa 2 (3 puntos)**

Hoja de borrador

Preguntas tipo Test (15 preguntas, 3 puntos)

1. Considerando las siguientes declaraciones qué valor tiene la expresión (cierto o falso):

```
int i=1,j =3;
float x = 5.2, y = 9.1;
```

...

Expresión: `(x < y) &&  (( i < j) | | (j > 8) )`

Valor: .....

2. ¿ Cuál es la salida de la siguiente código:?

...

```
int i, j = 0;
int vec1[5] = {0, 2, 4, 6, 8};
int  vec2[5] = {1, 3, 5, 7, 9};
for(i= 0; i<5; ++i)
       if(vec1[i] %vec2[i] ==0 )
              j += vec1[i] + vec2[i] ;
printf("%d", j);
```

...

Salida: .....

3. ¿ Cuál es la salida del siguiente código:?

```
int matriz[3][3] = {{3,2,1},{5,6,4},{9,8,7}};
calculo(matriz);
...
void calculo(int mat[][3])
{
   int i, j, s;

   for(i= 0; i<3; ++i)
   {
      s = 0;
      for(j = 0; j<3; ++j)
         if(mat[i][j] > s )
            s = mat[i][j];
   }
   printf("%d ", s);
   return 0;
}
```

Salida: .....

4. Considerando las siguientes declaraciones,escribir la respuesta de cada variable:

...

```
char u, v = ‘A’;
char *pu, *pv;
...
pv = &v;
*pv = v +2;
u = *pv + 3;
pu = &u;
```

...

Salida: u = ..., v = ..., *pu = ..., *pv = ...

5. ¿Cuál es la salida del siguiente código:?

```
typedef struct
{
   int valor;
   int codigo;
} Articulo;
...
int main()
{
   Articulo objeto;

   objeto.valor=2;
   objeto.codigo=3;
   funcion1(objeto);
   printf(“ %d %d” , objeto.valor, objeto.codigo ) ;
   funcion(&objeto);
   printf(“ %d %d” , objeto.valor, objeto.codigo ) ;
   return 0;
}

void funcion1( Articulo ficha)
{
   ficha.valor += 10 ;
   ficha.codigo = 3 * ficha.valor ;
}
void funcion ( Articulo *ficha)
{
   ficha -> valor += 10 ;
   ficha -> codigo = 3 * ficha -> valor ;
}
```

Salida: .....

6. ¿ Cuál es la salida del siguiente código:?

```
#include<stdio.h>
int func1( int n) ;
int main( )
{
   int n = 4 ;
   int res;
   res=func1(n);
   printf(“%d”,res);
   return 0;
}

int func1( int n )
{
   int s=1;
   if (n > 0 )
      s = n* func1(n – 1) ;
   return s;
}
```

Salida: .....

7. ¿Cuál es la salida de la siguiente código?:

si el archivo datos.txt contiene:

3

4

7

```
int main( )
{
   int n ;
   FILE *archivo ;
   int dato ;
   int sum_tot = 0 ;

   archivo= fopen("datos.txt", "r");
   ...
   n = fscanf(archivo, “ %d ", &dato);
   while (n == 1)
   {
      if(dato<6)
      sum_tot +=dato;
      n = fscanf(archivo, “ %d ", &dato);
   }
   printf("El valor de la suma es: %d\n", sum_tot);
   ...
}
```

Salida : .....

8. ¿Cuál es la salida de la siguiente código?:

```
#define N 20
...
int main(void)
{
   char palabra[N];
   int numero1;
   int numero2;
   float resultado;

   strcpy(palabra, “Esto es una prueba”);
   numero1 = strlen(palabra);
   strcpy(palabra, “Adios”);
   numero2 = strlen(palabra);
   resultado = numero1 / numero2;
   printf(“La división de la longitud es %f”, resultado);
   return 0;
}
```

Salida: .....

9. La función siguiente recibe dos vectores v1 y v2 de 5 elementos cada uno. Si los elementos de v1 son: 1 , 2 , 3 , 4 , 5 ¿Cuál es el valor devuelto por la función?

```
...
int funcion( int *v1 , int *v2 )
{
   int i , suma = 0 ;
   for ( i = 0 ; i< 5 ; ++i)
   {
      if (v1[i] %2 == 0)
         v2[i] = 2 * v1[i] ;
      else
         v2[i] = v1[i] ;
      suma += v2[i];
   }
   return suma ;
}
```

Valor devuelto por la función: .....

10. ¿Cuál es el valor final de los elementos del vector vec?

```
…
int vec[3]={1,2,3};
int i;
for(i=0;i<3;++i)
{
   switch(vec[i]) {
       case 1:
              vec[i]+=2;
              break;
       case 2:
              vec[i]=3;
              break;
       default:
              vec[i]=4;
   }
}
…
```

Vector: .....

11. Escribir el valor final de los elementos del vector vec.

```
…
int vec[4]={1,2,3,4};
int *p1;
int *p2;
int i;

p1=vec;
p2=p1+1;
for(i=0;i<2;++i)
{
   *p1=*p2;
   ++p1;
   ++p2;
}
…
```

Vector : .....

12. ¿Cuál es el valor final de los elementos del vector vec ?

…

```
int main()
{
   int vec[4]={1,3,5,7};
   calcular(vec);
   ...
}
void calcular(int *v)
{
   int i;
   int *punt;

   punt=v+3;
   for(i=0;i<4;++i)
   {
      v[i]=*punt;
      --punt;
   }
}
```

Vector: .....

13. Escribir el resultado de la ejecución del código siguiente:

…

```
typedef struct{
   int codigo;
   float precio;
}T_producto;

void calcular(T_producto *p);

int main()
{
   T_producto prod;
   T_producto *punt;
   prod.codigo=123;
   prod.precio=21.5;
   punt = &prod;
   calcular(punt);
   printf("El codigo=%d, el precio=%.1f",prod.codigo,prod.precio);
}

void calcular(T_producto *p)
{
   p->codigo=100;
   p->precio=10.5;
}
```

Salida .....

14. Completar el código siguiente. Utilizar la variable punt para escribir los datos de la estructura en la pantalla.

```
…
typedef struct{
   int edad;
   float peso;
}T_persona;

int main()
{
   T_persona persona;
   T_persona *punt;

   persona.edad=30;
   persona.peso=65.0;
……
……
……
}
```

15. ¿Qué hace esta función al recibir una cadena de caracteres?

```
int funcion( char *c)
{
   char *p ;
   p = c ;
   while( *p != '\0' )
       p++ ;
   return ( p – c ) ;
}
```

Respuesta: ……

Hoja de borrador

Códigos cortos (4 preguntas, 2 puntos)

1 Operadores

Escribir una función que reciba un valor entero y lo devuelva con las cifras al revés. Suponed que el valor siempre tiene 4 cifras. Ejemplo: si recibe 1234 debe devolver 4321, si recibe 12 debe devolver 2100

2 Estructuras

Escribe un `ejemplo` de declaraciones `typedef` y de definición de la variable `var`, para que las siguientes expresiones tengan sentido:

```
var[3].num_notas = 7;
var[5].nota[6] = 8.4;
var[0].fecha.dia=16;
```

3 Archivos de texto

Suponed que un archivo de texto contiene los siguientes datos de un conjunto de personas: Nombre (que es sólo el primer nombre, sin espacios), la edad (tipo `int`) y el peso (tipo `float`).

Escribe el código de la función `void Mostrar30(FILE *fp)`; que muestre por pantalla mediante `printf` todos los nombres de las personas con 30 años.

4 Archivos binarios

Un archivo binario contiene toda la información de los productos de una tienda. Se quiere disponer de una función que permita modificar el precio de un producto, indicando el número de registro del producto y el factor por el que se multiplica el precio. El prototipo de la función es:

```
void Actualizar(FILE *fp, int num_reg, float factor);
```

Por ejemplo, si el programa principal tiene abierto el archivo de productos y quiere aumentar un 20% el precio del producto número 24, se realiza la siguiente llamada:

```
Actualizar(fp,24,1.20);
```

Sabiendo que la estructura con que está construido el archivo es `t_producto`, se pide escribir el código de la función Actualizar.

```
typedef struct {
       char producto[N];
       float precio;
       int num_unidades;
} t_producto;
```

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!
```