Examen convocatoria de Febrero de 2002 — Modelo B

Nombre:_____

1. (2.5 puntos) Marca la respuesta verdadera de los conceptos que se señalan (Nota: los fallos descuentan puntos)

La diferencia entre acceso secuencial y acceso directo a un fichero es:

☐ En el primero los elementos se acceden mediante una secuencia, por ejemplo se leen dos, no se leen los dos siguientes etc y el directo lee siempre cada uno

☐ En el secuencial, necesariamente los datos deben ser de tipo texto con formato

☐ Que el acceso secuencial se realiza en el nivel lógico del fichero y el directo en el nivel físico

☐ En el secuencial para acceder a un dato hay que pasar necesariamente por todos los situados anteriormente en el fichero

Podemos recorrer completamente una matriz de dos dimensiones

☐ Usando una condicional múltiple con un bucle for en cada condición

☐ Usando un doble bucle for anidado

☐ Usando consecutivamente dos bucles for

☐ Usando un bucle while

La programación estructurada consiste en

☐ El uso de la metodología descendente, usando instrucciones de control de flujo iterativas y condicionales en la codificación

☐ Utilizar el concepto de ámbito de las variables, para definir la zona del programa donde se puede utilizar una variable

☐ El uso de todos los aspectos considerados en las dos afirmaciones anteriores

☐ El empleo de estructuras (`struct`) para el diseño de datos complejos

Un bucle for simple del tipo for (i=0; i< MAX; i++)

☐ Puede ser sustituido por un único bucle while haciendo los cambios que sean necesarios

☐ Puede ser sustituido por un condicional del tipo switch con los cambios que sean necesarios

☐ No puede ser sustituido por ninguna otra estructura de control de flujo

☐ Puede ser sustituido por dos bucles while anidados así while(i>0){ while(i<MAX){ i++;} }

Sea X una variable booleana (True o False). Entonces el resultado de X OR (NOT X) es

- [ ] Siempre False
- [ ] Siempre True
- [ ] NOT (X AND X)
- [ ] NOT (X OR X)

Dada una función llamada FUNC y un procedimiento llamado PROC

- [ ] FUNC y PROC pueden aparecer juntas en una expresión
- [ ] PROC puede formar parte de una instrucción de asignación
- [ ] PROC puede estar como operando en una expresión.
- [ ] FUNC puede estar como operando en una expresión

Las sentencias: `do{ x=100; while(x >0) {x = x-1;} } while(x<100);` **se ejecutarán**

- [ ] El `do..while` 1 vez y el `while` interno 100 veces
- [ ] El `do ... while` 100 veces y el `while` interno 1 vez
- [ ] El `do...while` 100 veces y el `while` interno 100 * 100 veces
- [ ] Infinitas veces

Una variable no inicializada

- [ ] Tiene que ser usada necesariamente como parte izquierda de una asignación para que el programa sea correcto
- [ ] Puede usarse en la parte derecha de una asignación si en la parte izquierda está ella misma
- [ ] Puede usarse como parte de una expresión
- [ ] puede existir en un programa ya que el compilador de C da error si no la inicializamos en la declaración.

Una calculadora de bolsillo (que hace sumas, restas, multiplicaciones y poco mas.) no es una máquina de Von Newmann porque

- [ ] Los algoritmos para hacer estas operaciones no están almacenados en memoria sino que forman parte de los circuitos de la calculadora
- [ ] Le falta la entrada de datos
- [ ] No tiene un lugar para almacenar los datos introducidos
- [ ] Tiene en memoria todos los algoritmos (para sumar, restar, multiplicar...) mientras que la Máquina de Von Newmann sólo puede tener un algoritmo

Si A es un *parámetro real* de una función denominada FUNC, quiere decir que:

☐ A sirve para codificar la función FUNC, es decir, aparece como variable en el código de FUNC

☐ Necesariamente A debe estar pasada por referencia, ya que tiene existencia real

☐ FUNC recibe en su llamada a A, perteneciendo A al módulo que llama a FUNC

☐ FUNC declara como una variable local a A, en la parte de declaración de variables de su codigo

2. (2 .5puntos)Encuentra los errores semánticos (de concepto, de uso, de ver si hace algo útil) en los siguientes códigos en C. Señálalos con un número y pon en la hoja de respuestas el número y la explicación. Puede haber más de un error en cada cuadro.

CODIGO 1

```
void cualquiercosa(int a, int *b)
{
 int aux;
 printf("Introduce dos enteros");
 scanf("%d  %d", &a, &b);
 if(a > *b){
   aux = a;
   a = *b;
   *b = aux;
 }
 while(*b > a){
   a = a + 1;
   *b = *b + 1;
 }
 printf("Resultado %d", *b);
}
```

CODIGO 2

```
int lee(char nombre []){
FILE *fp;
int resul, dato;
fp = fopen(nombre, "rb");
while(!feof(fp)){
   fread(&dato, sizeof(int), 1, fp);
   resul = resul + dato;
}
return(resul);
}
```

CODIGO 3

```
void rellena_numero(float num){
float aux;
aux = num;
printf("Escribe un numero real: ");
scanf("%f", &num);
}
main(){
float numero= 100.0;
int i=0;
do{
rellena_numero(numero);
if(numero < aux)
   printf("%f", numero);
}
while(numero != 59.36769);

}
```

CODIGO 4

```
//cuenta el numero de caracteres
//validos de una cadena
int cuenta_char(char cadena[ ]){
int num, i;
while(cadena[i] < '\0'){
  num++;
  i++;
}
printf("Numero de letras:%d",
num);
```

3. (2.5 puntos)

a)**Explica RAZONADAMENTE lo que hace el siguiente algoritmo: Nota: No me expliques**

```
void misterio(int N, char cadena[ ]){
//suponemos un tamaño de cadena suficientemente grande
int i, signo;

signo = N;
if(signo < 0)
   N = -N;
i = 0;
do{
   cadena[i] = N % 10 + '0';
   i++;
   N = N / 10;
} while(N > 0);
if(signo <0){
   cadena[i] = '-';
   i++;
}
cadena[i] = '\0';
}
```

lo que hace instrucción por instrucción.

b) Haz una traza del algoritmo, es decir indicando cómo se van transformando paso a paso los argumentos de "misterio", si la llamada es `misterio(-34567, cadena[ ]);`

4. (2.5 puntos)

a) Define los PROTOTIPOS de los siguientes subprogramas de la manera que te parezca más razonable, prestando especial atención al decidir los pasos (por valor o referencia) de los argumentos (si es que tiene argumentos). Tras escribir el prototipo JUSTIFICA TU ELECCION con una breve explicación. Puedes poner el nombre que quieras al subprograma.

NOTA: sabes que la forma de un prototipo es:

`<tipo de dato devuelto> <nombre subprograma> ( <tipo argumento> <nombre argumento,...);`

a.1 Un subprograma que sume dos numeros a y b en coma flotante y devuelva su resultado

a.2 Un subprograma que cargue un vector de enteros con valores validos hasta un determinado tamaño (se supone que el vector es mayor que dicho tamaño).

a.3 Un subprograma que reciba dos números enteros y devuelva dos valores: el resultado de la división entera y el resto de la división entera.

a.4 Un subprograma que saque por pantalla los diez primeros números pares.

a.5 Un subprograma que tome una cadena de caracteres y devuelva la longitud de la cadena y el número de veces que aparece la letra 'a'.

b) Elabora el diagrama de flujo del siguiente código

```
Z=1;
while(z < 3){
  if(a > b)
    if(c > d)
      if(e > f)
      X= 1;
      else
      X = 2;
    else
    X= 3;
  else
  X = 4;
  Z++; }
```