ENUNCIADOS PRÁCTICO ENTREGA 02.
ESTRUCTURAS DE DATOS

Ejercicio	Temática	Saber más
2.x.a.y	**Problemas generales C / C++**	Anexo 5. Resumen_C_para_practicas Anexo 6. Lenguaje C Anexo 7. Ejercicios resueltos C Anexo 8. Aprenda C++ Anexo 9. Sintaxis C++
2.x.b.y	**Estructuras estáticas**	Anexo 1. Programación de estructuras estáticas
2.x.c.y	**Estructuras dinámicas**	Anexo 2. Programación de estructuras dinámicas Anexo 3. Arboles_AVL
2.x.d.y	**Ficheros**	Anexo 4. Ficheros en C

Ejercicio	Temática	Videos / Tutoriales
2.x.a.y	**Problemas generales C / C++**	
2.x.b.y	**Estructuras estáticas**	https://www.youtube.com/watch?v=-Shr2s0gYao http://www.conclase.net/c/edd/index.php?cap=000#inicio
2.x.c.y	**Estructuras dinámicas**	https://www.youtube.com/watch?v=v0JLMmrlsOU http://www.conclase.net/c/edd/index.php?cap=000#inicio
2.x.d.y	**Ficheros**	https://www.youtube.com/watch?v=LT-EvYv9r9Y

Nota:

El código de cada ejercicio hace alusión a:

1: *Entrega 1*

x: *Nivel básico (1), intermedio (2) o avanzado (4)*

a-i: *Temática.*

y: *nº del ejercicio en orden correlativo dentro de los de su temática.*

Ejemplo: El ejercicio con código **1.1.c.1** *sería un ejercicio de la* ***Entrega 2*** *(Estructuras de datos), nivel* ***básico****; temática "Problemas generales C / C++";* ***primer ejercicio*** *de esa temática y nivel.*

2.1.a.1

1. Dadas las siguientes declaraciones de variables:

a. float
b. double
c. int
d. Ninguna de las respuestas anteriores es correcta.

2. La operación x && (y || z) devolverá:

a. Verdadero sólo en el caso de que **x, y, z** sean verdaderas.
b. Verdadero en el caso de que al menos una de ellas sea verdadera, independientemente del resto.
c. Falso, si por ejemplo x e y son falsas, independientemente del valor de z.
d. Ninguna de las respuestas anteriores es correcta

3. La diferencia fundamental entre un bucle while y un bucle do-while es ... :

a. Si la condición del bucle **while** se cumple dicho bucle deja de ejecutarse, mientras que si la condición del bucle **do-while** se cumple este bucle continúa.
b. Si la condición del bucle **while** se cumple dicho bucle continúa, mientras que si se cumple la condición del bucle **do-while** éste finaliza.
c. Un bucle while puede que no se llegue a ejecutar ni una sola vez, mientras que un bucle dowhile garantiza que al menos una vez son ejecutadas las sentencias que engloba.
d. Ninguna de las respuestas anteriores es correcta.

4. ¿La siguiente función calcula la suma de las 10 primeros números pares mayores de 100 que no son divisibles entre 3 ?

Si

5. Sea el siguiente trozo de código:

¿Cuál será el resultado que se mostrará en pantalla? **01233**

6. Dada la siguiente definición char cadena[ ]= "Hola mundo"

¿Cuál será el valor de la variable cadena[10]? **'\0'**

7. Dada la siguiente definición:

¿Qué doble bucle for muestra por pantalla 5 8 ?

a. for(int j=2;j<3;j++) for(int i=j-1;i<3;i++) cout<<t[i][j];
b. for(int j=1;j<3;j++) for(int i=j;i<3;i++) cout<<t[i][j];
c. for(int j=1;j<3;j++) for(int i=j-1;i<3;i++) cout<<t[i][j];
d. for(int j=1;j<3;j++) for(int i=j-1;i<=3;i++) cout<<t[i][j];
e. for(int j=0;j<3;j++) for(int i=j;i<3;i++) cout<<t[i][j];

8. Dadas la siguiente definición:

¿Qué instrucción permite acceder a la 'z' de Lopez? **cout << registro.cad3[4];**

9. Dado el siguiente trozo de código:

¿Cuántas veces se mostrará por pantalla Hola? **7 veces**

10. Dado el siguiente trozo de código:

¿Qué función implementa? **FACTORIAL**

2.2.a.2

1. Se define:

```
typedef struct (int dia , mes , anyo;) Fecha;
typedef struct (long dni; Fecha fecha_nacim;) Tp;
```

Indique cuál de las siguientes expresiones recoge por teclado el mes de nacimiento del registro apuntado por p:

a. scanf ("%d", *p.fecha_nacim.mes);
b. scanf ("%d", p->fecha_nacim.mes);
c. scanf ("%d", &p->fecha_nacim.mes);
d. scanf ("%d", p->&fecha_nacim.mes) ;

2. Dada la invocación

Podemos decir que el prototipo de este procedimiento es:

a. void hacer (int, float * , char [] , FILE *) ;
b. void hacer (int *, float *, char [], FILE *);
c. void hacer (int , *float, char * , FILE *);
d. void hacer (int, float &, char [], FILE *)

3. Dada la definición

Indique cuál de las siguientes sentencias permite acceder a la 'c' de García:

a. registro.cad3[3]
b. registro[3].cad3[3]
c. registro[5][3]
d. registro[3].cad3

4. Una función tiene que ordenar una tabla t de registros de tipo Treg por el campo c3. La sentencia de comparación entre dos elementos de la tabla dentro de la función es:

a. if (t[i]->c3 < t[j]->c3)
b. if (t[i]. c3 < t [j] , c3)
c. if (strcmp (t[i]. c3 , t[j] . c3) < 0)
d. if (*(t[i] . c3 < * (t[j] . c3))

5. Para detectar el final de una cadena de caracteres char s[10], que almacena la palabra [HOLA]

a. Almacena en el elemento 0 la longitud de la cadena, es decir, s [0] valdría 4.
b. Coloca un carácter '\0' después del último carácter utilizado, es decir, s[4] valdría '\0'.
c. Coloca un carácter '\0' en la última posición declarada de la cadena, es decir, s[9] valdría '\0'.
d. Ninguna de las anteriores.

6. Para acceder al número entero almacenado en la componente 3 de un vector de registros declarado como

se utilizaría

a. v[3] . int
b. v(3) . c1
c. v . c1 [3]
d. (v + 3) -> c1

7. Dado un fichero binario de registros de tipo Treg, la sentencia que posiciona el cursor de lectura preparado para leer el tercer registro es:

a. fseek (fdat, 3*sizeof(Treg), SEEK_ SET);
b. fseek (fdat, 2*sizeof(Treg), SET_SEEK);
c. fseek (fdat, 2*sizeof(Treg), SEEK_SET);
d. fseek (fdat, 3*sizeof(Treg), SEEK_CUR);

8. ¿Qué doble bucle for Imprime los elementos de una matriz n x n que se encuentren por debajo de la diagonal principal? Es decir dada `int t[3]= {1,2,3,4,5,6,7,8,9}` imprimiría 4 7 8.

a. for (i=0; i<n; i++) for (j=0, j<i; j++) printf ("%d", t[i] [j]);
b. for (i=0; i<n; i++) for (j=0, j<i+1; j++) printf ("%d", t[i] [j]);
c. for (i=0; i<n; i++) for (j=1, j<n; j++) printf ("%d", t[i] [j]);
d. for (i=0; i<n; i++) for (j=0, j<=i; j++) printf ("%d", t[i] [j]);

9. Se declara `char s[20];` se ejecuta la siguiente instrucción: `scanf("%s", s)`, se introduce por teclado: PEPITO GRILLO, e inmediatamente se ejecuta la instrucción: `puts(s);` ¿Cuál de los siguientes literales se visualizará en la pantalla?

a. PEPI
b. PEPITO GRILLO
c. PEPITO
d. PEPIT

10. Si se declara `float x, *p;` ¿Cuál de las siguientes expresiones es correcta?

a. p= &x;
b. x=p*;
c. &x=p;
d. &p=x;

11. Sea el siguiente código:

Si la table de enteros *tab* contiene los valores {0,0,0,0,0,0,0,0,0,0} y la variable entera *posición* tiene el valor 9, ¿cuál será el contenido de la tabla *tab* tras la llamada desplaza (*tab&posición*);?

a. { 1, 0, 0, 0, 0, 0, 0, 0, 0, 0,}
b. { 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,}
c. { 0, 0, 0, 0, 0, 0, 0, 0, 0, 1,}
d. { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,}

12. Si se quiere leer una variable c de tipo carácter de un fichero de texto denominado ftxt, una posible sentencia sería:

a. &c= fgetc (ftxt) ;
b. fscanf ("%c", &c, ftxt);
c. fscanf (ftxt, "%c", c);
d. fscanf (ftxt, "%c", &c);

2.2.a.3

1. Dadas las siguientes definiciones:

Indicar cuál de las siguientes llamadas es correcta:

a. func(pr1->c1, r1.c3, r1.c2);
b. func(r1.c1, r1.c3, r1.c2);
c. func(*r1.c1, pr1->c3, r1.c2);
d. func(&r1.c1, r1.c3, pr1->c2);

2. Para desplazar el cursor de L/E al principio de un archivo binario identificado mediante la variable pf la instrucción es:

a. fseek(pf, 0L, SEEK SET);
b. fseek(pf, 1L, SEEK SET);
c. fseek(pf, 0L, SEEK END);
d. ftell(pf);

3. Si se declara

¿Cuál de las siguientes expresiones es correcta?

a. p=&x;
b. x0 p*;
c. &x=p;
d. &p=x;

4. Sea el siguiente trozo de código:

El resultado por pantalla es el siguiente:

a. Da un error de compilación
b. 7 8 9 10 11 12 13 14 15
c. 7 8 9 10 11 12 13 14
d. 7.5 8.5 9.5 10.5 11.5 12.5 13.5 14.5

5. ¿Cuál es la región del mapa de memoria utilizando por un programa en C usado por las funciones de asignación dinámica de memoria?

a. La pila (stack).
b. El montón (heap).
c. La región donde se almacenan las variables globales.
d. La región donde se almacena el código del programa.

6. Si se declara

a. palabra [4] es el carácter 'a'
b. Da un error de compilación.
c. palabra [4] es el carácter '\0'
d. no se sabe el valor de palabra [4]

7. Indicar cuál de las siguientes afirmaciones es cierta:

a. Todos los campos de una estructura han de ser de tipos diferentes.

b. Un campo de una estructura puede ser un array bidimensional.
c. Un campo de una estructura no puede ser otra estructura.
d. Para leer una estructura de un fichero hay que hacerlo obligatoriamente campo a campo.

8. Dadas las siguientes instrucciones:

a. Ninguna de las dos es correcta.
b. Son exactamente lo mismo.
c. La primera reserva una matriz de 7 filas y 10 columnas.
d. La segunda reserva una matriz con un número de filas y columnas indeterminado.

9. La función:

a. Copia la cadena c2 a c1 machacando el contenido original de la cadena c1.
b. Concatena la cadena c2 a la cadena c1.
c. No está permitido incrementar las variables c1 y c2.
d. La última instrucción no es necesaria. Sin ella el programa hace exactamente lo mismo.

10. Dado el siguiente código:

Para mostrar el 20 deberemos utilizar la siguiente instrucción:

a. printf ("%d", *vector [2]);
b. printf ("%d", vector [2]);
c. printf ("%d", *(vector + 2));
d. Ninguna de las opciones anteriores es correcta.

11. Dadas las siguientes instrucciones:

a. Los valores almacenados en el vector después de estas instrucciones son: 2 2 3 4 4 6 7 8
b. Las instrucciones no son válidas.
c. Los valores almacenados en el vector después de estas instrucciones son: 2 2 3 4 5 5 7 8
d. Ninguna de las opciones anteriores es correcta.

12. Dado el siguiente trozo de código:

La salida sería:

a) 0 1 2 3 4 5
b) 4
c) 0 1 2 3 4
d) 5

2.1.b.1

1. Suponiendo que se ha declarado: int x[10], *p=x;

indique qué instrucciones pueden ser correctas

01) scanf("%d",&x[1]);
02) scanf("%d",x+1);
04) printf("%d",*x++);
08) scanf("%d",p++);

2. Dadas estas declaraciones: int a[5]={10,20,30,40,50}, *p=a, i=2;

01) printf("%d",*p[i]); imprime 30
02) printf("%d",*(p+i)); imprime 30
04) printf("%d",*++p+i); imprime 22
08) printf("%d",*a++); imprime 10

3. Con la siguiente declaración: int x[ ]={3,2,8,7,5};

01) La instrucción printf("%d", *x++); imprime 3
02) La instrucción printf("%d", (*x)++); imprime 3
04) La instrucción printf("%d", *(x+3)); imprime 7
08) La instrucción printf("%d", *(x+3)=-1); imprime -1

4. Con las siguientes instrucciones, indique qué afirmaciones son correctas:

01) printf("%d",*p->a); imprime 50
02) printf("%d",(++p)->a); imprime 51
04) printf("%s",++p->nom); imprime eptiembre
08) printf("%c",*++p->nom); imprime e

5. Con las siguientes declaraciones:

Indicar qué sentencia es **INCORRECTA**

01) scanf("%f %f", &x[0].real, &x[0].imag);
02) scanf("%f %f", p->real, p->imag);
04) scanf("%f %f", &p->real, &p->imag);
08) printf("%f %f", (*p).real, (*p).imag);

6. ¿Qué imprime este programa?

01) martesmiercolesjuevesviernessabadodomingo
02) lunesmartesmiercolesjuevesviernessabadodomingo
04) lunesartesmiercolesjuevesviernessabadodomingo
08) munesnartesoiercoleskueveswiernestabadoeomingo

7. Indicar qué sentencias hay que sustituir por el comentario para que el siguiente programa lea desde teclado todos los datos del array a.

01) scanf("%s %f", p[i]->nombre, p[i]->nota);
02) { gets(a[i].nombre); scanf("%f", &a[i].nota); }
04) { gets(pp[i]->nombre); scanf("%f", &pp[i]->nota); }

08) scanf("%s %f", &p[i]->nombre, &p[i]->nota);

8. Con las siguientes declaraciones, indique qué instrucciones pueden ser correctas:

01) pun=(struct ficha *)malloc(sizeof(struct ficha *));
02) pun=(struct ficha *)malloc(4*sizeof(struct ficha));
04) pun=(struct ficha *)calloc(3,sizeof(struct ficha));
08) pun=(struct ficha *)realloc(pun,5*sizeof(struct ficha));

9. La función:

01) Retorna el valor de x[i].
02) Retorna el valor de x[j].
04) Retorna la posición del mayor elemento del array x.
08) Retorna el valor del mayor elemento del array x.

10. En la siguiente función, indique qué instrucciones pueden ser correctas:

01) printf("%c", *cad++);
02) printf("%c", *(cad)++);
04) printf("%c", *(cad+3));
08) printf("%c", *(cad+3)='W');

11. ¿Qué imprime el programa siguiente?

01) Luis Sanchez, 7.50
02) Juan Garcia, 9.80
04) [valor basura], [valor basura]
08) Juan, 9.80

12. En el programa siguiente, dada la estructura struct alumno, indicar qué sentencias se han de sustituir por el COMENTARIO en la función leer_datos para que se lea desde teclado el nombre y la nota de un alumno.

01) strcpy(alu->nombre,nom);
02) alu->nombre=(char *)malloc((strlen(nom)+1));
if (alu->nombre==NULL) { puts("No hay memoria"); exit(0); }
strcpy(alu->nombre,nom);
04) gets(alu->nombre);
08) alu->nombre=(char *)malloc((strlen(nom)+1));
if (alu->nombre==NULL) { puts("No hay memoria"); exit(0); }
alu->nombre=nom;

2.2.b.2

1. Hacer una función con el siguiente prototipo: char *fun(char *cad);

Solución:

```
char *fun(char *cad)
{
       char *ini=cad;
       while(*cad)
       { if (*cad>='a' && *cad<="z") *cad+=('A'-'a');
              cad++;
       }
       return ini;
}
```

2. Escribir un programa que lea desde el teclado una frase compuesta por varias palabras en minúsculas (máximo 200 caracteres) y, con la ayuda de un puntero, la transforma para poner en mayúsculas solo la primera letra de cada palabra (si es una minúscula).

```
#include <stdio.h>
main( )
{ char cad[200], *p=cad; //el puntero p comienza desde el principio de cad
  printf("Teclea una frase: "); gets(cad);
  while (*p!=0) //mientras p no alcance el character \0 de final de cadena
  { if (p==cad || (*p!=' ' && *(p-1)==' ')) //condicion de primera letra de una palabra
    if (*p>='a' && *p<='z') //si es una letra minúscula
      *p = *p - ('a'-'A'); //también podría ser: *p -= 'a'-'A';
    p++; //p pasa a apuntar a la siguiente letra
  }
  printf("\nLa frase corregida es: %s", cad);
}
```

3. Crear un programa que reciba desde el teclado una fecha introducida en format numérico (25-12-2010) y la muestre en pantalla en formato texto (25 de diciembre de 2010).

```
#include <stdio.h>
int diasMes(int mm, int aa);
char *nombreMes(int mm);

main( )
{ struct tipofecha { int dia, mes, anyo; } fecha;
  int mal; //indica un error en los datos de entrada

do
{ printf("Introduce la fecha (dd-mm-aaaa): ");
  scanf("%2d-%2d-%4d", &fecha.dia, &fecha.mes, &fecha.anyo); //lee la fecha teclado
  mal = (fecha.dia<1 || fecha.dia>diasMes(fecha.mes, fecha.anyo)); //revisa valor dia
  mal = mal || (fecha.mes<1 || fecha.mes >12); //revisa el valor del mes
  mal = mal || (fecha.anyo<1000 || fecha.anyo>3000); //revisa el valor del año
} while (mal); //si algo está mal, pide la fecha de nuevo

printf("La fecha es: %d de %s de %d", fecha.dia, nombreMes(fecha.mes), fecha.anyo);
}


int diasMes(int mm, int aa) //retorna el n° de dias del mes mm
{
  int d=0;
  switch (mm)
  { case 1: case 3: case 5: case 7: case 8: case 10: case 12: //meses con 31 dias
      d = 31; break;
    case 4: case 6: case 9: case 11: // meses con 30 dias
```

```
      d = 30; break;
    case 2: //Febrero
      if (aa%4==0 && aa%100!=0 || aa%400==0) //formula para saber si un año es bisiesto
        d = 29;
      else d = 28;
  }
  return d;
}


char *nombreMes(int mm) //retorna el nombre del mes mm
{
  char *mes[ ] = {"Mes incorrecto", "enero", "febrero", "marzo", "abril",
"mayo","junio","julio", "agosto", "septiembre", "octubre", "noviembre", "diciembre"};
  return (mm>0 && mm<13) ? mes[mm] : mes[0]; //retorna el punter correcto
}
```

4 Crear una función con prototipo void maymin(float *p, float *mayor, float *menor, int n); que calcule el valor mayor y el menor de los elementos de un array de números de tipo float.

```
void maymin(float *p, float *mayor, float *menor, int n)
{
  *mayor=*menor=*p++;
  while(--n>0)
  {
    if (*mayor<*p) *mayor=*p;
    if (*menor>*p) *menor=*p;
    p++;
  }
}
```

5. Crear una función con prototipo char *fun(char *s); que permita hacer un duplicado de una cadena de caracteres dada por s.

```
char *fun(char *s)
{
  char *p=(char *)malloc(strlen(s)+1);
  strcpy(p,s);
  return p;
}
```

6. Diseñar una función fun() que reciba como argumentos de entrada dos cadenas de caracteres cad1 y cad2, y que genere otra cadena que contenga la concatenación de aquellas dos (cad1 seguida de cad2).

```
char *fun(char *cad1, char *cad2)
{
  char *aux; int tam;
  tam = strlen(cad1)+strlen(cad2)+1; // +1 para el caracter \0 final
  aux = (char *)malloc(tam); //Crear la nueva cadena
  if (aux!=NULL)
    { strcpy(aux,cad1); strcat(aux,cad2); } //Copiar cad1+cad2 en aux
  return aux;
}
```

7. Con la siguiente declaración:

```
tficha *fun(int n)
{
  tficha *pun;
  pun = (tficha *)malloc(sizeof(tficha)*n);
  if (pun==NULL) printf("No hay memoria.");
  return pun;
}
```

8. Con las siguientes declaraciones:

```
char *mayor(struct ficha *x, int n)
{
  int i=0,m=0;
  for (i=1; i<n; i++)
  if (x[i].nota>x[m].nota)
    m=i;
  return x[m].nom;
}
```

9. Hacer un programa con argumentos en línea de órdenes que imprima por pantalla los argumentos que se le pasan pero en letras mayúsculas.

```
main(int argc, char **argv)
{
  while(*++argv != NULL)
  printf("%s ", strupr(*argv));
}
```

Otra solución:

```
main(int argc, char *argv[ ])
{ int i;
  for (i=1; i<argc; i++)
  printf("%s ", strupr(argv[i]));
}
```

2.2.c.1

2. Una lista enlazada está formada por estructuras con el siguiente formato:

01) **q=p->sig;**
p->sig=(tipo *)malloc(sizeof(tipo));
p->sig->num=50; p->sig->sig=q;

02) p->sig=(tipo *)malloc(sizeof(tipo));
p->sig->num=50; p->sig->sig=p->sig;

04) **q=p->sig;**
p->sig=(tipo *)malloc(sizeof(tipo));
p=p->sig; p->num=50; p->sig=q;

08) **q=(tipo *)malloc(sizeof(tipo));**
***q=*p; q->num=50; p->sig=q;**

3. La función fun busca un elemento en una lista lineal enlazada, en la que cada elemento tiene el siguiente tipo de datos: typedef struct nodo

```
01)   while((p != NULL) && (sw == 0))
      if( p->dato==num) sw=1; else p++;
      if (sw) return(p); else return NULL;
02)   while((p != NULL) && (sw == 0))
      if( p->dato==num) sw=1; else p=p->sig;
      if (sw) return(p); else return NULL;
04)   while((p->sig != NULL) && (sw == 0))
      if( p->dato==num) sw=1; else p=p->sig;
      if (sw) return(p); else return NULL;
08)   while((p != NULL) && (sw == 0))
      if( p->dato==num) sw=1; else p=p->sig;
      if (sw) return(--p); else return NULL;
```

2. La pila de la figura es controlada mediante el puntero de ámbito global pi.

```
01)   strcpy(nom,pi->nombre); printf("%s\n ",nom);
      pt=pi; pi=pi->sig;
02)   printf("%s\n ",pi->nombre);
      pi=pi->sig; pt=pi;
04)   nom = pi->nombre; printf("%s\n ", nom);
      pt=pi; pi=pi->sig;
08)   strcpy(nom,pi->nombre);
      pt=pi; pi=pi->sig;
      printf("%s\n ",nom);
```

3. La cola de la figura es controlada mediante dos punteros de ámbito global: pi y pf.

```
01)   strcpy(pt->nombre ,cad); pt->sig= NULL;
      if (pf==NULL) pi=pt; else pf->sig=pt;
      pf=pt;
02)   pt->nombre=cad) ;
      if (pf==NULL) pi=pt; else pf->sig=pt;
      pt->sig= NULL; pf=pt;
04)   strcpy(pt->nombre ,cad) ;
      if (pi==NULL) pf=pt; else pi->sig=pt;
      pt->sig= NULL; pi=pt;
08)   strcpy(pt->nombre ,cad) ;
      if (pf==NULL) pi=pt; else pf->sig=pt;
      pt->sig= NULL; pf=pt;
```

4. En una pila se han ido almacenando los siguientes números por orden de llegada:

01) Con la siguiente operación pop obtendremos 15.
02) Con la siguiente operación pop obtendremos 15 y con la siguiente pop obtendremos 20.
04) Con la siguiente operación pop obtendremos 12.
08) Con la siguiente operación pop obtendremos 12 y con la siguiente pop obtendremos 20.

5. En una cola se han ido almacenando los siguientes números por orden de llegada:

01) Con la siguiente operación pop obtendremos 15.
02) Con la siguiente operación pop obtendremos 15 y con la siguiente pop obtendremos 20.

04) Con la siguiente operación pop obtendremos 12.
08) Con la siguiente operación pop obtendremos 12 y con la siguiente pop obtendremos 20.

3. Si creamos un árbol binario de búsqueda con las cadenas siguientes: uno dos tres cuatro cinco seis siete ocho nueve

Ninguna es correcta

5. Si creamos un árbol binario de búsqueda que almacena claves de tipo cadenas de caracteres, y con las siguientes cadenas introducidas en este orden:

01) la altura o profundidad del árbol es 4
02) un recorrido en Postorden es: rosa verde negro marron blanco naranja amarillo azul rojo
04) "blanco" es un nodo terminal
08) un recorrido en Inorden es: verde rosa rojo negro naranja marron blanco azul amarillo

7. Se ha creado un árbol binario ordenado formado por estructuras del siguiente tipo:

01) 2 3 4 5 6 7 8 9 12
02) 8 7 12 6 2 3 4 5 9
04)12 9 8 7 6 5 4 3 2
08)4 5 3 2 6 7 8 9 12

2.3.c.2

1. Una lista enlazada está formada por estructuras con la siguiente definición: struct pp

```
void fun(struct pp *p)
{
  struct pp *aux;
  aux=p->sig; *p=*aux;
  free(aux);
}
```

2. Una lista enlazada está formada por estructuras del tipo nodo:

```
float fun (nodo * p)
{
  float suma = 0; int cont = 0;
  while (p != NULL)
  { suma += p->dato; cont ++; p = p->sig; }
  return (suma/cont);
}
```

3. Existe en el disco un fichero de nombre datos.dat que contiene registros desordenados de tipo registro en los que el campo clave no se repite.

```
*/ COMENTARIO-1 */
if (r!=NULL)
{ imprimir(r->izqdo);
  printf("Clave: %d, Nombre: %s\n", r->datosreg.clave, r->datosreg.nombre);
  imprimir(r->dcho);
}

/* COMENTARIO-2 */
if (r!=NULL)
{ borrararbol(r->izqdo); borrararbol(r->dcho); free(r); }

/* COMENTARIO-3 */
if ((pf=fopen("datos.dat", "r+"))==NULL)
{ printf("Fichero datos.dat no existe\n"); exit(0); }
  while (fread(&dato, sizeof(registro), 1, pf)==1)
  {
    raiz = anadir(dato, raiz);
  }
  fclose(pf);
```

2.2.d.1

1. Se tiene en el disco un fichero de nombre t1.txt que contiene únicamente los 4 caracteres siguientes: velo

01) velo
02) udkn
04) wfmp
08) elo

2. Suponiendo que se ha declarado: FILE *pf;

```
01)     if (pf=fopen("datos.dat","a")) { puts("error"); exit(0); }
02)     char nom[ ]="datos.dat";
        if (!(pf=fopen(nom,"w")) { puts("error"); exit(0); }
04)     char nom[ ]="datos.dat", x[ ]="a";
        if (!(pf=fopen(nom,x)) { puts("error"); exit(0); }
08)     char *nom="datos.dat";
        if (!(pf=fopen(&nom[0],"a")) { puts("error"); exit(0); }
```

3. Si ejecutamos el siguiente programa:

01) Hay que introducir el nombre del archivo que se quiere crear y cuyo contenido va a ser el caracter A
02) No se puede abrir el archivo
04) Hay que introducir el nombre del archivo que se quiere crear y luego los caracteres a grabar en binario
08) Da error de compilación

4. Con las siguientes declaraciones:

01) fgets("Una línea",39,pf);
02) fgets(cad+1,30,pf);
04) fputs("Una línea",pf);
08) fputs("Una línea",20,pf);

5. Un fichero que contiene estructuras de tipo struct pp acaba de ser abierto para lectura y es referenciado mediante un puntero pf. Si x es una variable de ese tipo de estructuras

01) fseek(pf,2,SEEK_SET); fread(&x,sizeof(struct pp),1,pf);
02) fseek(pf,1,SEEK_SET); fread(&x,sizeof(struct pp),1,pf);
04) fseek(pf,2*sizeof(struct pp),SEEK_SET); fread(&x,sizeof(struct pp),1,pf);
08) fseek(pf,1*sizeof(struct pp),SEEK_SET); fread(&x,sizeof(struct pp),1,pf);

6. Un fichero cuyo puntero de referencia es FILE *pf; ha sido abierto en modalidad de lectura y está trabajando con estructuras de tipo struct pp.

Si se ha declarado **struct pp x[10];**

01) fread(x,sizeof(struct pp),1,pf); lee un registro y lo mete en x[0].
02) fread(x+3,sizeof(struct pp),1,pf); lee un registro y lo mete en x[3].
04) fread(&x[3],sizeof(struct pp),1,pf); lee un registro y lo mete en x[3].
08) rewind(pf); printf("%ld",ftell(pf)); imprime el numero de bytes del fichero.

7. Dadas estas declaraciones:

01) fwrite(x,sizeof(x),1,p);
02) fwrite(x,sizeof(struct pp),40,p);
04) fwrite(&x,sizeof(x),1,p);
08) fwrite(&x[0],sizeof(x),1,p);

8. ¿Qué imprime el siguiente programa, suponiendo que el fichero de texto fich.txt existe y que contiene un texto de 100 caracteres?

01) 100
02) 99
04) Nada, da error de compilación
08) 101

9. En un fichero de tipo base de datos abierto mediante el puntero pf y formado por 100 registros de tipo struct reg

01) fseek(pf, 0, SEEK_END); fread(&dato, sizeof(struct reg), 1, pf);
02) fseek(pf, 0, SEEK_END); fread(&dato, sizeof(dato), 1, pf);
04) fseek(pf, 99*sizeof(struct reg), SEEK_SET); fread(&dato, sizeof(struct reg), 1, pf);
08) fseek(pf, 100*sizeof(dato), SEEK_SET); fread(&dato, sizeof(dato), 1, pf);

2.2.d.2

1. Diseñar una función con el siguiente prototipo int fun(char *nom);

```
int fun(char *nom)
{
  int con=0; char c;
  FILE *p=fopen(nom,"r");
  while ((c=fgetc(p))!=EOF)
    { con++; putchar(c); }
  fclose(p);
  return con;
}
```

O BIEN:

```
int fun(char *nom)
{
  int con=0; char c;
  FILE *p=fopen(nom,"r");
  c=fgetc(p);
  while(!feof(p) && !ferror(p))
    { con++; printf("%c",c); c=fgetc(p); }
  fclose(p);
  return con;
}
```

2. Crear un programa que genere en disco un archivo de texto encriptado de nombre "mifi.txt" cuyo contenido se va introduciendo caracter a caracter desde el teclado.

```
main()
{
  char c;
  FILE *pf;
  pf=fopen("mifi.txt","w");
  if (pf==NULL) { printf("Error de acceso a fichero.\n"); exit(0); }
  printf("Introduce texto terminando con Ctrl+Z: ");
  while((c=getchar())!=EOF && !ferror(pf))
    fputc(c+1, pf);
  if (ferror(pf)) printf("Error al grabar el fichero.\n");
  fclose(pf);
}
```

Otra solución:

```
main()
{
  char c;
  FILE *pf;
  pf=fopen("mifi.txt","w");
  if (pf==NULL) { printf("Error de acceso a fichero.\n"); exit(0); }
  printf("Introduce texto terminando con Ctrl+Z: ");
  c = getche();
  while(!feof(stdin) && !ferror(pf))
    { fputc(c+1, pf); c = getche(); }
  if (ferror(pf)) printf("Error al grabar el fichero.\n");
  fclose(pf);
}
```

3. Realizar un programa que cree un fichero de texto cuyo nombre es el primer argumento de la línea de órdenes y cuyo contenido son una serie de líneas separadas por caracteres \n que contienen cada una el resto de los argumentos de la línea de órdenes, escritos en mayúsculas.

```
int main(int argc, char *argv[ ])
{ int i,j;
  FILE *pf=fopen(argv[0],"w+");
  if (pf==NULL) { printf("Error de creación del fichero.\n"); exit(0); }
  for (i=1; i<argc; i++)
  { for (j=0; argv[i][j]!=0; j++)
    if (argv[i][j]>='a' && argv[i][j]<='z')
      argv[i][j]+=('A'-'a');
    fputs(argv[i], pf); fputc('\n', pf);
  }
  fclose(pf);
  printf("El fichero %s ha sido creado.\n", argv[0]);
}
```

4. Se ha creado un fichero usando la función fwrite(), que almacena estructuras del siguiente tipo: struct pp

```
struct pp fun(FILE *pun)
{
  struct pp una={"",0}; int n;
  printf("Numero de registro a leer: "); scanf("%d",&n);
  fseek(pun,sizeof(struct pp)*(n-1),SEEK_SET);
  fread(&una,sizeof(struct pp),1,pun);
  if (feof(pun)) printf("Registro no existe.\n");
  return una;
}
```

5. Escribir una función llamada fun con el prototipo indicado, que escriba carácter a carácter la cadena de caracteres apuntada por el puntero cad, sobre el fichero abierto en modo escritura y apuntado por el puntero pf, sin utilizar ninguna variable adicional aparte de cad y pf.

```
void fun(char *cad, FILE *pf)
{
  while (*cad!=0 && !ferror(pf))
  { fputc(*cad, pf); cad++; }
}
```