Ejercicios a realizar durante las clases

Ejercicios simples con el objetivo de que se prueben características concretas de C que les serán útiles para la realización de las prácticas. Estos procedimientos/funciones se prueban en un programa principal realizados por los alumnos.

1.- Implementar un procedimiento con un parámetro de tipo puntero a entero que muestre el valor de la variable. **(acceso a la variable anónima)**

```
Void func(int *pa) {
        Printf("%d\n",*pa);
}
```

2.- Implementar un procedimiento con dos parámetros de tipo puntero a entero. Modificar su valor y mostrarlo por pantalla. **(Uso de parámetros de E/S)**

```
Void func(int *pa,int *pb) {
        Int temp;
        Temp=*pa;
        *pa=*pb
        *pb=temp;
        Printf("%d %d\n",*pa,*pb);
}
```

3.- Implementar un procedimiento que tenga como parámetro un puntero a una zona de memoria de números enteros e introduzca valores leídos por teclado. Definir el tamaño de la zona en el programa principal e invocar a este procedimiento**. (Uso de aritmética de punteros/uso arrays)**

```
Void func(int *pa,int nelem) {
        Int I;
        For (i=0;i<nelem;i++)
                Scanf("%d",&pa[i]);
}
```

```
Main() {
Int nelem;
Int *arr;
Scanf("%d",&nelem);
Arr=malloc(sizeof(int)*nelem);
If (arr==NULL)
        Perror("Error en peticion de memoria");
Else {
        Func(arr,nelem);
        }
}
```

4.- Realizar lo mismo definiendo en el programa principal un array. **(Uso de aritmética de punteros/uso arrays)**

La función no varía

```
Main() {
Int arr[NELEM];          // #define NELEM 10
Func(arr,nelem);
}
```

5. Implementar un procedimiento con un parámetro de tipo puntero a una zona de memoria de números enteros y mostrar su contenido. **(Uso de aritmética de punteros/uso arrays)**

```
Void func(int *pa,int nelem) {
        Int I;
        For (i=0;i<nelem;i++)
                Printf("%d %d\n",pa[i]);
```

```
}
```

6.- Implementar un procedimiento con un parámetro de tipo puntero a una zona de memoria de números enteros y modificar su contenido. **(Uso de aritmética de punteros/uso arrays)**

```
Void func(int *pa,int nelem) {
        Int I;
        For (i=0;i<nelem;i++)
                Pa[i]++;
}
```

7.- Definir un registro compuesto por dos registros. El primero de ellos (Info1) formado por los campos nombre y apellidos y el segundo (Info2) por los campos edad y altura. Definir dos procedimientos que tengan como parámetro un puntero a Info1 y a Info2 respectivamente y que introduzcan valores a la estructura. **(Uso de registros y punteros)**

```
Struct Info1 {
 Char nombre[30];
 Char apellidos[80];
};

Struct Info2 {
 Int edad,altura;
};

Void func1(struct info1 *pinfo1) {
        Printf("Introduzca nombre: ");
        Scanf("%s",pinfo1->nombre);
```

Printf(“Introduzca apellidos: “);

Scanf(“%s”,pinfo1->apellidos);

}

Void func2(struct info2 *pinfo2) {

Printf(“Introduzca edad: “);

Scanf(“%d”,&(pinfo2->nombre));

Printf(“Introduzca altura: “);

Scanf(“%d”,&(pinfo2->altura));

}

8.- Implementar la siguiente biblioteca **(Primer ejemplo de manejo de listas dinámicas y manejo de ficheros)**:

```
#ifndef __LISTA_H__
#define __LISTA_H__


typedef struct T_Nodo* T_Lista;

struct T_Nodo {
                    unsigned num;
                    T_Lista sig;
             };

// Crea la estructura utilizada

       void Crear(T_Lista* lista);

// Destruye la estructura utilizada.
       void Destruir(T_Lista* lista);

// Rellenar lista
void Rellenar (T_Lista *lista);

void  Mostrar (T_Lista lista);

void EscribirF(char *nombre,T_Lista lista);

void LeerDeFichero(char*nombre,T_Lista *l);


#endif /* __LISTA_H__ */


////////////////////////////////////////////////////////////////
SOLUCION:

#include <stdio.h>
```

```c
#include <stdlib.h>

typedef struct T_Nodo* T_Lista;

struct T_Nodo {
                    unsigned num;
                    T_Lista sig;
             };


// Crea la estructura utilizada

       void Crear(T_Lista* lista) {
              *lista=NULL;
       }

// Destruye la estructura utilizada.
       void Destruir(T_Lista* lista) {
              T_Lista sig;
              while (*lista != NULL) {
                     sig=(*lista)->sig;
                     free((void *)*lista);
                     *lista=sig;
              }
       }

// Rellenar lista
void Rellenar (T_Lista *lista) {
// No dice que hay que hacer
int i=0;
T_Lista nuevo;

for (i=0;i<100;i++) {
  nuevo=(T_Lista)malloc(sizeof(struct T_Nodo));     // Comprobar si no NULL
  nuevo->num=i;
  nuevo->sig=*lista;
  *lista=nuevo;
  }

}

void  Mostrar (T_Lista lista) {
       while (lista != NULL) {
              printf("%d ",lista->num);
              lista=lista->sig;
       }
}


void EscribirF(char *nombre,T_Lista lista) {
       FILE *fd;

       if ((fd=fopen(nombre,"wt"))==NULL) {
              perror("Error en creacion de fichero");
       }
       else {
       while (lista != NULL) {
              fprintf(fd,"%d ",lista->num);
              lista=lista->sig;
```

```
        }
      fclose(fd);
      }

}

void LeerDeFichero(char*nombre,T_Lista *l) {
      FILE *fd;
      int num,leidos;
      T_Lista nuevo;

      if ((fd=fopen(nombre,"rt"))==NULL) {
                  perror("Error en apertura de fichero");
            }
            else {

            while (!feof(fd)) {
                  leidos=fscanf(fd,"%d",&num);
                  if (leidos>0) {

                        // Estaria bien operacion de insertar numero
                        nuevo=(T_Lista)malloc(sizeof(struct T_Nodo));//
Comprobar si no NULL
                        nuevo->num=num;
                     nuevo->sig=*l;
                        *l=nuevo;
                  }

              }
            fclose(fd);
            }

}



int main(int argc,char *argv[]) {
      T_Lista l,l2;

      Crear(&l);
      Crear(&l2);

      Rellenar(&l);
      Mostrar(l);
      EscribirF("p1.txt",l);
      printf("\n");
      LeerDeFichero("p1.txt",&l2);
      Mostrar(l2);
      Destruir(&l);
      Destruir(&l2);
      return 0;
}
```

9.- Definir una biblioteca para gestión de una lista dinámica de enteros con las operaciones habituales: Crear, Mostrar, Insertar por el principio, Insertar por el final, Insertar ordenado, Eliminar un elemento, Destruir **(Segundo ejemplo de manejo de listas dinámicas)**: