Programación en C/C++

Hoja de problemas 3 :
“Funciones, ficheros, y otros aspectos de C”

David Rozas Domingo
I.T.I.Sistemas

Desarrolle programas en C para los siguientes problemas propuestos.

Nota 1 : El compilador gcc da una advertencia indicando que main debe devolver algo. Por ello en todos los programas main devuelve un int (0 en caso de ejecución correcta).

Nota 2 : En el ejercicio 12 he tenido problemas para que me reconociera las funciones que se definen en el archivo cabecera (pila.h). Así que se incluye tanto la versión que presenta fallos en la construcción (pila.c, pila.h y ejercicio12.c), como una versión en la que se incluye todo en un mismo archivo (ejercicio12bis.c).

3)

```
/************************************************************
      Asignatura : Programacion en C/C++
      Practica : Hoja de problemas 3
      Autor : David Rozas Domingo
************************************************************/

/* Descripcion : Ejercicio 3. Sustituye los caracteres en blanco
                        de una cadena por guiones, y los cuenta*/

#include <stdio.h>
#include <string.h>

int sustituir (char cad[]);

int main(void)
{


      char str1[100];
      int cont;

      printf("Escriba una frase (max 100 caracteres) : ");
      gets(str1);

      printf("Cadena original: \"%s\"\n", str1);
      cont = sustituir(str1);
      printf("Cadena sustituida : \"%s\"\n", str1);
      printf("Se han realizado %i sustituciones\n", cont);

      return 0;
}

/* Sustituye los espacios en blanco por guiones en un string*/
int sustituir (char cad[])
{
      int i,cont;

      for(i=0, cont=0; i<strlen(cad); i++)
            if(cad[i] == ' ')
            {
                  cad[i]= '-';
                  cont++;
            }
      return cont;
}
```

4)

```
/************************************************************
      Asignatura : Programacion en C/C++
      Practica : Hoja de problemas 3
      Autor : David Rozas Domingo
************************************************************/

/* Descripcion : Ejercicio 4. Recibe una palabra, y determina
                        si es un palindromo*/

#include <stdio.h>
#include <string.h>

int es_palindromo (char cad[]);

int main(void)
{
      char str1[30];

      printf("Escriba una palabra : ");
      gets(str1);


      if(es_palindromo(str1))
            printf("\"%s\" es una palabra palindroma", str1);
      else
            printf("\"%s\" no es una palabra palindroma", str1);

      return 0;
}

/*Determina si una cadena es un palindromo*/
int es_palindromo (char cad[])
{
      int izq,drcha,salir;
      izq = 0;
      drcha = strlen(cad) - 1;
      salir = 0;

      /*Recorreremos la palabra comparando por los extremos*/
      while((izq<=drcha) && !salir)
      {
            if(cad[izq] != cad[drcha])
                  salir = 1;
            izq++;
            drcha--;
      }

      return !salir;

}
```

5)

```
/************************************************************
      Asignatura : Programacion en C/C++
      Practica : Hoja de problemas 3
      Autor : David Rozas Domingo
************************************************************/

/* Descripcion : Ejercicio 5. Se lee una cadena por pantalla y se almacena
```

```
                        cada caracter en una lista. Posteriormente se eliminan
todos
                        los espacios en blanco de dicha lista*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>

/*Definiciones para tipo lista*/
struct tipoNodoCaracter{
      char car;
      struct tipoNodoCaracter *sig;
      };
typedef struct tipoNodoCaracter tNodoCar;

/*Cabeceras de funciones auxiliares*/
void insertar (tNodoCar **lista, char car);
void recorrer (tNodoCar *lista);
void eliminar_espacios (tNodoCar *lista);
void liberar (tNodoCar *lista);

int main(void)
{
      char frase[50];
      tNodoCar *lista;
      int i;

      lista = NULL;

      printf("Introduce una frase (max. 50 car.): ");
      gets(frase);


      for(i=0; i<strlen(frase); i++)
      {
            insertar(&lista, frase[i]);
      }

      printf("\nCotenido inicial de la lista : ");
      recorrer(lista);

      printf("\nCotenido posterior de la lista : ");
      eliminar_espacios(lista);
      recorrer(lista);
      liberar(lista);

      return 0;
}


/* Funciones auxiliares                                                  */
/************************************************************/

/*Inserta un nodo al final de la lista */
void insertar (tNodoCar **lista, char car)
{
      tNodoCar *p1, *p2;

      p1 = *lista;

      /*Si es el primer elemento*/
      if(p1==NULL)
      {
            p1 = malloc(sizeof(tNodoCar));
```

```
            if (p1 != NULL)
            {
                  /*Guardamos info, y hacemos q ese nodo apunte a null*/
                  p1->car = car;
                  p1->sig = NULL;
                  *lista = p1;
            }

      }else{
            /*Si habia mas elementos */

            /*Vamos hacia el ultimo*/
            while(p1->sig!=NULL)
                  p1 = p1->sig;

            p2 = malloc(sizeof(tNodoCar));

            if(p2 != NULL)
            {
                  /*Guardamos info en nuevo nodo, hacemos que apunte a null*/
                  p2->car = car;
                  p2->sig = NULL;

                  /*Y lo apuntamos desde el que anteriormente era el ultimo*/
                  p1->sig = p2;
            }
      }

}

/*Recorre la lista mostrando su contenido por la salida estandar*/
void recorrer (tNodoCar *lista)
{
      while(lista != NULL)
      {
            printf("%c", lista->car);
            lista = lista->sig;
      }
}

/*Recorre la lista eliminando los nodos cuyo contenido
  es un caracter en blanco*/
void eliminar_espacios (tNodoCar *lista)
{
      tNodoCar *pAux;

      pAux = NULL;

      while(lista->sig != NULL)
      {
            /*Si el nodo proximo contiene caracter en blanco*/
            if((lista->sig)->car == ' ')
            {
                  /*Lo borramos, y apuntamos desde nuestro nodo actual al que
apuntaba*/
                  pAux = lista->sig;
                  lista->sig = pAux->sig;
                  free(pAux);
            }else{
                  /*Si no, seguimos recorriendo la lista*/
                  lista = lista->sig;
            }
```

```
	}
}

/*Libera todos los nodos de la lista*/
void liberar(tNodoCar *lista)
{
	tNodoCar *pAux;

	while(lista != NULL)
	{
		pAux = lista;
		lista = lista->sig;
		free(pAux);
	}

}
```

6)

```
/************************************************************
	Asignatura : Programacion en C/C++
	Practica : Hoja de problemas 3
	Autor : David Rozas Domingo
************************************************************/

/* Descripcion : Ejercicio 6. Decide si un caracter aparece
			en una cadena de caracteres*/

#include <stdio.h>
#include <string.h>


#define N_CAR 100

int main(void)
{
	char str1[N_CAR];
	char car;
	int pos[N_CAR], i, j;/*el vector pos almacena las posiciones donde
aparece*/

	printf("Escriba una frase : ");
	gets(str1);

	printf("Escriba un caracter : ");
	scanf("%c", &car);

	for(i=0, j=0; i<strlen(str1); i++)
	{
		if(str1[i]==car)
		{
			pos[j] = i;
			j++;
		}
	}

	if (j>0)
	{
		printf("Se ha encontrado el caracter %c %i veces, en las posiciones
: \n", car, j);
		for(i=0; i<j;i++)
			printf("%i , ", pos[i]);
	}else{
		printf("El caracter %c no se encuentra en la palabra\n", car);
```

```
	}

	return 0;
}
```

7)

```
/************************************************************
	Asignatura : Programacion en C/C++
	Practica : Hoja de problemas 3
	Autor : David Rozas Domingo
************************************************************/

/* Descripcion : Ejercicio 7. Contiene estructuras para almacenar polinomios
                        y permite que se realicen sumas de los mismos*/

#include <stdio.h>
#include <stdlib.h>

/*Definiciones para tipo lista*/
struct tipoNodoPolinomio{
	int coeficiente;
	unsigned int exponente;
	struct tipoNodoPolinomio *sig;
	};
typedef struct tipoNodoPolinomio tNodoPolinomio;

/*Cabeceras de funciones auxiliares*/
void insertar (tNodoPolinomio **lista, int coeficiente, int exponente);
void recorrer (tNodoPolinomio*lista);
void liberar (tNodoPolinomio *lista);
void crearPolinomio (tNodoPolinomio **polinomio);
void sumarPolinomios (tNodoPolinomio **pol1, tNodoPolinomio *pol2);

int main(void)
{
	tNodoPolinomio *pol1,*pol2;

	pol1 = NULL;
	pol2 = NULL;

	crearPolinomio(&pol1);
	crearPolinomio(&pol2);

	printf("\nPolinomio 1 : ");
	recorrer(pol1);
	printf("\nPolinomio 2 : ");
	recorrer(pol2);

	printf("\nSuma de ambos : ");
	sumarPolinomios(&pol1,pol2);
	recorrer(pol1);

	liberar(pol1);
	liberar(pol2);

	return 0;
}


/* Funciones auxiliares                                              */
/*********************************************************/

/*Crea un polinomio */
```

```
void crearPolinomio (tNodoPolinomio **polinomio)
{
	int j,i,coefAux;
	unsigned int expAux;

	printf("¿Cuantos terminos tiene su polinomio? : ");
	scanf("%i", &i);

	for (j=0; j<i; j++)
	{
		printf("Coeficiente del termino %i : ", j+1);
		scanf("%i", &coefAux);
		printf("Exponente del termino %i : ", j+1);
		scanf("%i", &expAux);

		insertar(polinomio,coefAux,expAux);
	}

}


/*Suma los valores de dos polinomios. El resultado se devuelve en el primero*/
void sumarPolinomios (tNodoPolinomio **pol1, tNodoPolinomio *pol2)
{

	while (pol2 != NULL)
	{
		insertar(pol1,pol2->coeficiente,pol2->exponente);
		pol2 = pol2->sig;
	}
}


/*Inserta un nodo al final de la lista*/
void insertar (tNodoPolinomio **lista, int coeficiente, int exponente)
{
	tNodoPolinomio *p1, *p2, *p3;

	p1 = *lista;
	int encontrado;

	/*Si es el primer elemento*/
	if(p1==NULL)
	{
		p1 = malloc(sizeof(tNodoPolinomio));

		if (p1 != NULL)
		{
			/*Guardamos info, y hacemos q ese nodo apunte a null*/
			p1->coeficiente = coeficiente;
			p1->exponente = exponente;
			p1->sig = NULL;
			*lista = p1;
		}

	}else{
		/*Si habia mas elementos */

		/*Primero recorremos, para ver si ya existe ese coeficiente, y en
ese caso sumarlo*/
		encontrado = 0;
		p3 = p1;
		while((p3 != NULL) && (!encontrado))
		{
```

```
                  if(p3->exponente==exponente)
                  {
                        p3->coeficiente += coeficiente;
                        encontrado = 1;
                  }else{
                        p3 = p3->sig;
                  }
            }

            /*Si no existia, lo guardamos en un nuevo nodo al final*/
            if (!encontrado){
                  /*Vamos hacia el ultimo*/
                  while(p1->sig!=NULL)
                  p1 = p1->sig;

                  p2 = malloc(sizeof(tNodoPolinomio));

                  if(p2 != NULL)
                  {
                        /*Guardamos info en nuevo nodo, y hacemos que apunte a
null*/
                        p2->coeficiente = coeficiente;
                        p2->exponente = exponente;
                        p2->sig = NULL;

                        /*Y lo apuntamos desde el que anteriormente era el
ultimo*/
                        p1->sig = p2;
                  }
            }


      }

}


/*Recorre la lista mostrando su contenido por la salida estandar*/
void recorrer (tNodoPolinomio *lista)
{
      while(lista != NULL)
      {
            if(lista->sig == NULL)
                  if (lista->exponente>0)
                        printf("%ix^%i", lista->coeficiente, lista->exponente);
                  else
                        printf("%i", lista->coeficiente);
            else
                  if (lista->exponente>0)
                        printf("%ix^%i + ", lista->coeficiente, lista-
>exponente);
                  else
                        printf("%i + ", lista->coeficiente);

            lista = lista->sig;
      }
}


/*Libera todos los nodos de la lista*/
void liberar(tNodoPolinomio *lista)
{
      tNodoPolinomio *pAux;
```

```
	while(lista != NULL)
	{
		pAux = lista;
		lista = lista->sig;
		free(pAux);
	}

}
```

8)

```
/************************************************************
	Asignatura : Programacion en C/C++
	Practica : Hoja de problemas 3
	Autor : David Rozas Domingo
************************************************************/

/* Descripcion : Ejercicio 8. Suma dos matrices de tamaño dinamico*/

#include <stdio.h>
#include <stdlib.h>

struct tipoNodo{
	int valor;
	struct tipoNodo *sig;
	};
typedef struct tipoNodo tNodo;

/*Cabeceras de funciones auxiliares*/
int leerValorPos();
void insertar (tNodo **mat, int val);
void recorrer (tNodo *mat, int nfil);
void sumarMatrices (tNodo *mat1, tNodo *mat2);
void liberar(tNodo *mat);

int main(void)
{

	int nfil1,nfil2,ncol1,ncol2;
	tNodo *m1,*m2;
	int i,j,valAux;

	m1 = NULL;
	m2 = NULL;

	/*Leemos los valores de filas y columnas*/
	printf("Numero de filas de la primera matriz? \n");
	nfil1 = leerValorPos();
	printf("Numero de columnas de la primera matriz? \n");
	ncol1 = leerValorPos();
	printf("Numero de filas de la segunda matriz? \n");
	nfil2 = leerValorPos();
	printf("Numero de columnas de la segunda matriz? \n");
	ncol2 = leerValorPos();

	/*Comprobamos que se puedan sumar*/
	if((nfil1==nfil2)&&(ncol1==ncol2))
	{
		/*Leemos los valores de las matrices*/
		for(i=0; i<nfil1; i++)
			for(j=0; j<ncol1;j++)
			{
				printf("Escriba el valor para la pos [%i,%i] de la
```

```
matriz 1: ", i,j);
                        scanf("%i", &valAux);
                        insertar(&m1,valAux);
                  }

            for(i=0; i<nfil1; i++)
                  for(j=0; j<ncol1;j++)
                  {
                        printf("Escriba el valor para la pos [%i,%i] de la
matriz 2: ", i,j);
                        scanf("%i", &valAux);
                        insertar(&m2,valAux);
                  }

            /*Mostramos su contenido*/
            printf("\tMatriz 1 \n");
            recorrer(m1,nfil1);
            printf("Matriz 2 \n");
            recorrer(m2,nfil1);

            /*Realizamos y mostramos la suma*/
            sumarMatrices(m1,m2);
            printf("Matriz 1 + Matriz 2 \n");
            recorrer(m1,nfil1);

            /*Y por ultimo liberamos la memoria*/
            liberar(m1);
            liberar(m2);

      }else{
            printf("Las matrices tienen distintas dimensiones. No se pueden
sumar\n");
      }

      return 0;
}

/* Funciones auxiliares                                                    */
/***************************************************/

/*Devuelve un valor positivo para filas o columnas*/
int leerValorPos()
{
      int val;

      do
      {
            printf("Introduce un valor : ");
            scanf("%i", &val);

            if (val<=0)
                  printf("Tiene que ser un valor positivo \n");

      }while(val<=0);

      return val;
}

/*Inserta un nodo al final de la lista */
void insertar (tNodo **mat, int valor)
{
      tNodo *p1, *p2;

      p1 = *mat;
```

```
	/*Si es el primer elemento*/
	if(p1==NULL)
	{
		p1 = malloc(sizeof(tNodo));

		if (p1 != NULL)
		{
			/*Guardamos info, y hacemos q ese nodo apunte a null*/
			p1->valor = valor;
			p1->sig = NULL;
			*mat = p1;
		}

	}else{
		/*Si habia mas elementos */

		/*Vamos hacia el ultimo*/
		while(p1->sig!=NULL)
			p1 = p1->sig;

		p2 = malloc(sizeof(tNodo));

		if(p2 != NULL)
		{
			/*Guardamos info en nuevo nodo, hacemos que apunte a null*/
			p2->valor = valor;
			p2->sig = NULL;

			/*Y lo apuntamos desde el que anteriormente era el ultimo*/
			p1->sig = p2;
		}
	}
}

/*Recorre la matriz mostrando su contenido por la salida estandar*/
void recorrer (tNodo *mat, int nfil)
{
	int nfilAux = 1;
	printf("\t");
	while(mat != NULL)
	{
		printf(" %i ", mat->valor);
		if ((nfilAux % nfil)==0)
			printf("\n\t");

		nfilAux++;
		mat = mat->sig;
	}
}

/*Suma dos matrices. La suma se guarda en la primera de ellas*/
void sumarMatrices (tNodo *mat1, tNodo *mat2)
{
	while(mat1!=NULL)
	{
		mat1->valor += mat2->valor;
		mat1 = mat1->sig;
		mat2 = mat2->sig;
	}
}

/*Libera todos los nodos de la matriz*/
void liberar(tNodo *mat)
```

```
{
	tNodo *pAux;

	while(mat!= NULL)
	{
		pAux = mat;
		mat = mat->sig;
		free(pAux);
	}
}
```

9)

```
/************************************************************
	Asignatura : Programacion en C/C++
	Practica : Hoja de problemas 3
	Autor : David Rozas Domingo
************************************************************/

/* Descripcion : Ejercicio 9. Lee dos ficheros de entradas con enteros.
			Los guarda ordenadamente en una estructura dinamica, y
			los escribe en un nuevo fichero*/

#include <stdio.h>
#include <stdlib.h>

struct tipoNodo{
	int valor;
	struct tipoNodo *sig;
	};
typedef struct tipoNodo tNodo;

/*Cabeceras de funciones auxiliares*/
void insertar_ordenado (tNodo **mat, int val);
void guardar (tNodo *mat);
void liberar(tNodo *mat);


int main(void)
{
	FILE *fichero_origen1 = fopen("fent1.dat", "rb");
	FILE *fichero_origen2 = fopen("fent2.dat", "rb");
	tNodo *lista;
	int dato;

	lista = NULL;

	/*Almacenamos ordenadamente en nuestra lista el contenido de ambos
ficheros*/
	while(fread(&dato, sizeof(dato), 1, fichero_origen1)==1)
		insertar_ordenado(&lista, dato);

	while(fread(&dato, sizeof(dato), 1, fichero_origen2)==1)
		insertar_ordenado(&lista, dato);

	/*Lo guardamos en un fichero, y liberamos la estructura*/
	guardar(lista);
	liberar(lista);

	fclose(fichero_origen1);
	fclose(fichero_origen2);

	return 0;
```

```
}


/* Funciones auxiliares                                                  */
/************************************************************/

/*Inserta un nodo en funcion de su valor en la lista */
void insertar_ordenado(tNodo **lista, int valor)
{
      tNodo *pActual, *pNuevo, *pAnterior;

      pActual = *lista;
      pAnterior = *lista;

      /*Buscamos su posicion*/
      while((pActual !=NULL) && (pActual->valor <valor))
      {
            pAnterior = pActual;
            pActual = pActual->sig;
      }

      /*Reservamos memoria para el nuevo nodo*/
      pNuevo = malloc(sizeof(tNodo));
      if (pNuevo!=NULL)
      {
            pNuevo->valor = valor;

            /*Y enlazamos*/
            if(pAnterior==NULL || pAnterior==pActual)
            {
                  /*Insertamos al principio*/
                  pNuevo->sig = pAnterior;
                  *lista = pNuevo;
            }else{
                  /*Insertamos entre medias o al final*/
                  pNuevo->sig = pActual;
                  pAnterior->sig = pNuevo;
            }
      }

}

/*Recorre la lista guardando su contenido en un fichero de texto*/
void guardar (tNodo *lista)
{
      FILE *fichero_destino = fopen("fsal.txt", "wt");

      int i = 1;
      while(lista != NULL)
      {
            if (i%3 != 0)
                  fprintf(fichero_destino,"%i ", lista->valor);
            else
                  fprintf(fichero_destino,"%i\n", lista->valor);

            i++;
            lista = lista->sig;
      }

      fclose(fichero_destino);
}


/*Libera todos los nodos de la matriz*/
```

```
void liberar(tNodo *lista)
{
	tNodo *pAux;

	while(lista!= NULL)
	{
		pAux = lista;
		lista = lista->sig;
		free(pAux);
	}
}
```

10)

```
/************************************************************
	Asignatura : Programacion en C/C++
	Practica : Hoja de problemas 3
	Autor : David Rozas Domingo
************************************************************/

/* Descripcion : Ejercicio 10. Procesa un texto guardando las lineas en
			una estructura dinamica, contando las vocales, etc.*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

struct tipoNodo{
	struct tipoNodo *sig;
	char car;
	};
typedef struct tipoNodo tNodo;

void recorrer (tNodo *lista);
void insertar (tNodo **lista, char car);
void liberar(tNodo *lista);
void inicializar_vocales(int vocales[]);


int main(void)
{
	FILE *fichero = fopen("prueba.txt", "rt");
	int n;
	tNodo *lista;
	lista = NULL;
	int vocales[5];

	int i;

	if (fichero!=NULL)
	{
		while(!feof(fichero))
		{
			n = fgetc(fichero);

			if (n != EOF)
			{
				insertar(&lista, n);
			}

		}
		fclose(fichero);

		/*Recorremos mostrando la info*/
```

```
            recorrer(lista);
            /*Por ultimo liberamos la memoria*/
            liberar(lista);

      }else{
            printf("Error al abrir el fichero \n");
      }

      return 0;
}


/*Inserta un nodo al final de la lista */
void insertar (tNodo **lista, char car)
{
      tNodo *p1, *p2;

      p1 = *lista;

      /*Si es el primer elemento*/
      if(p1==NULL)
      {
            p1 = malloc(sizeof(tNodo));

            if (p1 != NULL)
            {
                  /*Guardamos info, y hacemos q ese nodo apunte a null*/
                  p1->car = car;
                  p1->sig = NULL;
                  *lista = p1;
            }

      }else{
            /*Si habia mas elementos */

            /*Vamos hacia el ultimo*/
            while(p1->sig!=NULL)
                  p1 = p1->sig;

            p2 = malloc(sizeof(tNodo));

            if(p2 != NULL)
            {
                  /*Guardamos info en nuevo nodo, hacemos que apunte a null*/
                  p2->car = car;
                  p2->sig = NULL;

                  /*Y lo apuntamos desde el que anteriormente era el ultimo*/
                  p1->sig = p2;
            }
      }
}


/*Recorre la lista mostrando su contenido por la salida estandar*/
void recorrer (tNodo *lista)
{
      char car;
      int nA,nE,nI,nO,nU;
      int numLineas = 1;
      int vocales[5];

      inicializar_vocales(vocales);
      printf("%i) ",numLineas);
```

```
	while(lista != NULL)
	{
		car = lista->car;

		if (car=='\n')
		{
			printf("%c", car);
			printf("num. a = : %i ", vocales[0]);
			printf("num. e = : %i ", vocales[1]);
			printf("num. i = : %i ", vocales[2]);
			printf("num. o = : %i ", vocales[3]);
			printf("num. u = : %i ", vocales[4]);

			inicializar_vocales(vocales);
			numLineas ++;

			/*Pintamos el indice de linea, excepto para la ultima*/
			if(lista->sig != NULL)
				printf("\n%i) ", numLineas);

		}else{

			printf("%c",car);
			switch (car){
			 case 'a':
					vocales[0]++;
					break;
			 case 'A':
					vocales[0]++;
					break;
			 case 'e':
					vocales[1]++;
					break;
			  case 'E':
					vocales[1]++;
					break;
			 case 'i':
					vocales[2]++;
					break;
			 case 'I':
					vocales[2]++;
					break;
			 case 'o':
					vocales[3]++;
					break;
			 case 'O':
					vocales[3]++;
					break;
			 case 'u':
					vocales[4]++;
					break;
			 case 'U':
					vocales[4]++;
			 }
			}
		lista = lista->sig;
	}
}

/*Libera todos los nodos de la lista*/
void liberar(tNodo *lista)
{
	tNodo *pAux;
```

```
	while(lista != NULL)
	{
		pAux = lista;
		lista = lista->sig;
		free(pAux);
	}

}

/*Inicializa los valores del array que cuenta
  el numero de vocales*/
void inicializar_vocales(int vocales[])
{
	int i;

	for(i=0; i<5; i++)
	{
		vocales[i] = 0;
	}
}
```

11)

```
/************************************************************
	Asignatura : Programacion en C/C++
	Practica : Hoja de problemas 3
	Autor : David Rozas Domingo
************************************************************/

/* Descripcion : Ejercicio 11. Procesa un archivo de texto calculando los
			valores y devuelve sus resultados en un fichero
binario*/

#include <stdio.h>
#include <stdlib.h>

int main(void)
{
	FILE *fichero_origen = fopen("datos.txt", "rt");
	int n1, n2, res;
	char signo;
	FILE *fichero_destino = fopen("resultados.dat", "wb");

	/*Leemos del fichero de texto */
	while(fscanf(fichero_origen, "%i %c %i",&n1, &signo, &n2) != EOF)
	{
		/*Realizamos y guardamos el resultado de la operacion adecuada en
funcion del signo*/
		if (signo=='+')
		{
			res = n1 + n2;
			fwrite(&res, sizeof(res), 1, fichero_destino);
		}
		else if (signo == '-')
		{
			res = n1 - n2;
			fwrite(&res, sizeof(res), 1, fichero_destino);
		}
		else
		{
			printf("Caracter de signo desconocido. No se guardara en
fichero binario.\n");
		}
	}
```

```
	fclose(fichero_destino);
	fclose(fichero_origen);

	return 0;
}
```

12)

[ejercicio12.c]

```
/************************************************************
	Asignatura : Programacion en C/C++
	Practica : Hoja de problemas 3
	Autor : David Rozas Domingo
************************************************************/

/* Descripcion : Ejercicio 12. Utiliza el TAD pila,  realizando
                         inserciones y extracciones del mismo*/

#include <stdio.h>
#include <stdlib.h>
#include "pila.h"


int main(void)
{
	tNodo *mi_pila;
	int valAux;

	CrearVacia(&mi_pila);
	InsertarElemento(&mi_pila, 1);
	InsertarElemento(&mi_pila, 2);
	InsertarElemento(&mi_pila, 3);
	valAux = DevolverCima(&mi_pila);
	printf("Extrayendo valor %i \n", valAux);
	valAux = DevolverCima(&mi_pila);
	printf("Extrayendo valor %i \n", valAux);
	valAux = DevolverCima(&mi_pila);
	printf("Extrayendo valor %i \n", valAux);

	return 0;
}
```

[pila.h]

```
/************************************************************
	Asignatura : Programacion en C/C++
	Practica : Hoja de problemas 3
	Autor : David Rozas Domingo
************************************************************/

/* Descripcion : Ejercicio 12. Fichero de cabecera para el TAD pila*/

#include <stdio.h>
#include <stdlib.h>

struct tipoNodo{
	int valor;
	struct tipoNodo *sig;
	};
typedef struct tipoNodo tNodo;
```

```
void CrearVacia (tNodo **pila);
void InsertarElemento(tNodo **pila, int valor);
int DevolverCima(tNodo **pila);
```

[pila.h]

```
/*************************************************************
      Asignatura : Programacion en C/C++
      Practica : Hoja de problemas 3
      Autor : David Rozas Domingo
*************************************************************/

/* Descripcion : Ejercicio 12. Fichero de definicion para el TAD pila*/


/*Inicializa una pila*/
void CrearVacia (tNodo **pila)
{
      *pila = NULL;
}

/*Inserta un elemento en la cima de la pila*/
void InsertarElemento(tNodo **pila, int valor)
{
      tNodo *pAux;

      if(pila==NULL)
      {
            /*Si es el primer elemento*/
            pAux = malloc(sizeof(tNodo));
            if (pAux!=NULL)
            {
                  /*Guardamos info, y le apuntamos con el indice ppal*/
                  pAux->valor = valor;
                  pAux->sig = NULL;
                  *pila = pAux;
            }
      }else{
            /*Si habia mas elementos*/
            pAux = malloc(sizeof(tNodo));
            if (pAux!=NULL)
            {
                  /*Guardamos info, apuntamos al contenido anterior, y le
apuntamos desde ppal.*/
                  pAux->valor = valor;
                  pAux->sig = *pila;
                  *pila = pAux;
            }
      }
}

/*Devuelve el elemento que esta en la cima, y lo elimina de la pila*/
int DevolverCima(tNodo **pila)
{
      tNodo *pAux;
      int valAux;

      pAux = *pila;
      if(pAux==NULL)
      {
            printf("¡La pila está vacia!\n");
            return 0;
```

```
	}else{
		valAux = pAux->valor;
		/*Desplazamos el indice ppal, y liberamos la memoria*/
		*pila = pAux->sig;
		free(pAux);
		return valAux;
	}
}
```

[ejercicio12bis.c]

```
/*************************************************************
	Asignatura : Programacion en C/C++
	Practica : Hoja de problemas 3
	Autor : David Rozas Domingo
*************************************************************/

/* Descripcion : Ejercicio 12bis. Implementa el TAD pila y lo utiliza
                         desde el programa principal*/

#include <stdio.h>
#include <stdlib.h>

struct tipoNodo{
	int valor;
	struct tipoNodo *sig;
	};
typedef struct tipoNodo tNodo;

void CrearVacia (tNodo **pila);
void InsertarElemento(tNodo **pila, int valor);
int DevolverCima(tNodo **pila);


int main(void)
{
	tNodo *mi_pila;
	int valAux;

	CrearVacia(&mi_pila);
	InsertarElemento(&mi_pila, 1);
	InsertarElemento(&mi_pila, 2);
	InsertarElemento(&mi_pila, 3);
	valAux = DevolverCima(&mi_pila);
	printf("Extrayendo valor %i \n", valAux);
	valAux = DevolverCima(&mi_pila);
	printf("Extrayendo valor %i \n", valAux);
	valAux = DevolverCima(&mi_pila);
	printf("Extrayendo valor %i \n", valAux);

	return 0;
}
/*Inicializa una pila*/
void CrearVacia (tNodo **pila)
{
	*pila = NULL;
}

/*Inserta un elemento en la cima de la pila*/
void InsertarElemento(tNodo **pila, int valor)
{
	tNodo *pAux;
```

```
	if(pila==NULL)
	{
		/*Si es el primer elemento*/
		pAux = malloc(sizeof(tNodo));
		if (pAux!=NULL)
		{
			/*Guardamos info, y le apuntamos con el indice ppal*/
			pAux->valor = valor;
			pAux->sig = NULL;
			*pila = pAux;
		}
	}else{
		/*Si habia mas elementos*/
		pAux = malloc(sizeof(tNodo));
		if (pAux!=NULL)
		{
			/*Guardamos info, apuntamos al contenido anterior, y le
apuntamos desde ppal.*/
			pAux->valor = valor;
			pAux->sig = *pila;
			*pila = pAux;
		}
	}
}

/*Devuelve el elemento que esta en la cima, y lo elimina de la pila*/
int DevolverCima(tNodo **pila)
{
	tNodo *pAux;
	int valAux;

	pAux = *pila;
	if(pAux==NULL)
	{
		printf("¡La pila está vacia!\n");
		return 0;
	}else{
		valAux = pAux->valor;
		/*Desplazamos el indice ppal, y liberamos la memoria*/
		*pila = pAux->sig;
		free(pAux);
		return valAux;
	}
}
```