

# Ejercicios del Tema 5

## Punteros

2017-2018

### 1. Ejercicios resueltos

#### 1.1. Cambio de dos variables

Ejemplo de una función que, mediante el paso por referencia, intercambia el valor de dos variables.

```
#include <stdio.h>

void cambio (int *p, int *q);

int main()
{
    int a = 1, b = 7;

    printf("a vale %d y b vale %d\n", a, b);
    // Función que intercambia el valor de dos variables, accediendo a
    // su dirección de memoria dado que usa el paso por referencia (no
    // entrega ningún resultado)
    cambio(&a, &b);

    printf("a vale %d y b vale %d\n", a, b);

    return 0;
}

void cambio (int *p, int *q)
{
    int aux;
    aux = *p;
    *p = *q;
    *q = aux;
}
```

#### 1.2. Ecuación de segundo grado

```
#include <stdio.h>
#include <math.h> //para la funcion raiz cuadrada sqrt

// Funcion que combina paso por valor (a, b , c) y referencia (r1 y r2)
int raices(float a, float b, float c, float *r1, float *r2);

int main()
{
    float a, b, c; //coeficientes
    float r1, r2; //raices, soluciones
    int ret;
```

```

printf("Introduzca los coeficientes a, b y c.\n");
printf("de ax2 + bx + c = 0, separados por espacios.\n");

scanf("%f %f %f", &a, &b, &c);

printf("Ecuacion %f x2 + %f x + %f = 0\n",a,b,c);

ret = raices(a, b, c, &r1, &r2);

switch(ret)
{
case 0: printf("Soluciones reales %f %f\n", r1, r2);
break;
case 1: printf("Raiz doble %f\n", r1);
break;
case 2: printf("Raices complejas conjugadas %f+-i%f\n", r1, r2);
break;
case 3: printf("Ecuacion lineal, raiz: %f\n", r1);
break;
case 4: printf("Error parametros\n");
break;
}
}
// Los datos iniciales se pasan por valor (lectura), y los resultados
// por referencia (escritura)
int raices(float a, float b, float c, float *r1, float *r2)
{
if(a != 0.0) //ecuacion cuadrada
{
float disc;//discriminante
disc = b * b - 4 * a * c;
if(disc > 0)//dos soluciones reales
{
*r1 = (-b + sqrt(disc)) / 2 * a;
*r2 = (-b - sqrt(disc)) / 2 * a;
return 0;
}
else if(disc == 0)//raiz doble
{
*r1 = *r2 = -b / 2 * a;
return 1;
}
else//soluciones complejas
{
*r1 = -b / 2 * a;
*r2 = sqrt(-disc) / 2 * a;
return 2;
}
}
else //ecuacion no cuadrada
{
if(b != 0.0)
{
*r1 = -c / b;
return 3;
}
else
return 4;
}
}
}

```

## 2. Ejercicios propuestos

### 2.1. Longitud de una cadena de caracteres

Diseñe un programa que calcule la longitud de una cadena de caracteres usando una función basada en punteros.

```
int stringLength(char *string);
```

### 2.2. Copia de cadenas de caracteres

Diseñe un programa que copie dos cadenas de caracteres usando una función basada en punteros. Esta función imita el comportamiento de la función strcpy de la librería string.h (véase el capítulo 4, diapositiva 40).

```
void copyString(char *to, char *from);
```

### 2.3. Suma de un vector

Diseñe un programa que calcule la suma de un vector usando una función basada en punteros.

```
int sumaVector(int *vector, int n);
```

### 2.4. Asignación dinámica de memoria

Escribe un programa que permita al usuario rellenar un vector de dimensión variable. En primer lugar, el programa pregunta al usuario el número de elementos que tendrá el vector. A continuación, empleando asignación dinámica de memoria, el programa reservará memoria para este vector, y solicitará al usuario los valores del mismo. Finalmente, el programa mostrará el contenido de este vector.

### 2.5. Media, máximo y mínimo de un vector de datos

Escribe un programa que calcule el valor medio, el valor máximo y el valor mínimo de los datos contenidos en un vector de números reales empleando punteros. La función de cálculo debe acomodarse al siguiente prototipo:

```
void stats(float *vector, int n, float *media, float *max, float *min);
```

Debe escribir dos versiones diferentes. La primera versión trabajará con un vector de dimensión fija y conocida (p.ej. 5 elementos). La segunda versión debe emplear asignación dinámica de memoria, siguiendo el mismo itinerario que el ejercicio anterior.

*Nota: Este ejercicio es equivalente al ejercicio 6 del capítulo 4 (vectores). En aquel caso se usa notación de vectores, y ahora se debe emplear notación de punteros. Además, allí se usan tres funciones independientes, y ahora se debe usar una única función que entrega los tres resultados.*

### 2.6. Movimiento de un punto (estructura)

Construye un programa que sirva para mover un punto contenido en el plano (usando un tipo de datos punto) usando funciones basada en punteros a la estructura punto. El programa funciona con la interacción del usuario a partir de las teclas 'w' (arriba), 'a' (izquierda), 'x' (abajo), 'd' (derecha), y sus correspondientes mayúsculas. Por ejemplo, si el usuario aprieta la tecla 'w' el punto se desplaza hacia arriba una unidad, y si aprieta 'W' se desplaza cinco unidades hacia arriba. El programa muestra en pantalla las coordenadas del punto tras cada interacción del usuario. El programa termina cuando el usuario pulsa la tecla 'q'.

```
// Función para controlar el desplazamiento horizontal, siendo xy el
// punto, y n el número de unidades a desplazar
void horizontal(punto *xy, int n);
// Función para controlar el desplazamiento vertical
void vertical(punto *xy, int n);
```