Punteros a función

Cuando se programa en C se disponen de punteros a varios tipos de datos (char*, int*, float*, double*) que permiten hacer muchas cosas: reservar memoria dinámica, crear referencias y alias, etc.
Ahora bien, también es posible crear punteros que apunten a funciones como si fueran cualquier otro tipo de datos.
Los ejemplos muestran cómo declarar punteros a funciones y cómo utilizarlos de varias formas distintas.

Ejemplo básico

```c
 #include <stdio.h>
//Funcion de ejemplo
void funcion()
{
	printf("Se ha entrado en la funcion\n");
}
int main(int argc,char *argv[])
{

	//Creamos el puntero a la funcion funcion
	void (*puntero_funcion)() = &funcion;

	//Llamamos la funcion a traves del puntero
	puntero_funcion();

	return 0;
}
```

Como se puede ver, no tiene demasiada diferencia con el resto de los punteros.

Con parámetros

Si la función a la que se quiere apuntar recibe algún parámetro, hay que indicar sus tipos al declarar el puntero. Por ejemplo: la función recibe dos enteros y muestra sus valores.

Nótese que cuándo se declara el puntero se escriben los tipos de los argumentos entre paréntesis:

```c
#include <stdio.h>

//Funcion que recibe dos enteros
void funcion(int valor1, int valor2)
{
	printf("Valor 1 = %d\n", valor1);
	printf("Valor 2 = %d\n", valor2);
}

int main(int argc,char *argv[])
```

```
{
	//Se crea el puntero a la funcion. Hay que indicar el tipo de los
	//parametros que recibe.
	void (*puntero_funcion)(int, int) = &funcion;

	//Se llama la funcion a traves del puntero
	puntero_funcion(2, 5);

	return 0;
}
```

Función como parámetro a otra función

Las funciones también pueden pasarse como parámetros a otras funciones. En este código se define una función principal que recibe como parámetros dos números a, b y un puntero a una función entera. La función principal pasa a y b como parámetros a la función del puntero y escribe el valor que devuelve por pantalla:

```
#include <stdio.h>

int sumar(int a, int b)
{
	return a + b;
}

int restar(int a, int b)
{
	return a - b;
}

void funcion_principal(int a, int b, int (*funcion)(int, int)){

	int resultado = funcion(a, b);
	printf("El resultado es %d\n", resultado);
}

int main(int argc,char *argv[])
{
	//Se definen dos valores enteros cualesquiera
	int num1 = 5;
	int num2 = 4;

	//Se invoca la funcion principal, pasandole la funcion de SUMA
	printf("\nSuma:\n");
	funcion_principal(num1, num2, sumar);

	//Se invoca la funcion principal, pasandole la funcion de RESTA
	printf("Resta:\n");
	funcion_principal(num1, num2, restar);

	return 0;
}
```

Con este ejemplo puede verse que los punteros a funciones sirven para eliminar la redundancia en el código: al pasar la función como parámetro se ahorra tener que escribir variantes de la función principal para realizar cada una de las operaciones.