Miscelánea

1. Lectura de caracteres dentro de un bucle

Cuando se emplea la función `scanf` para leer caracteres de forma repetida (por ejemplo, dentro de un bucle `for` o `while`), hay que escribir un espacio delante de `%c`. Este espacio sirve para descartar la tecla intro que pulsamos después de cada carácter:

```
scanf(" %c")
```

2. Colores

Para escribir letras con colores hay que emplear los denominados «ANSI escape codes». En StackOverflow se puede encontrar el siguiente ejemplo:

```
#include <stdio.h>

#define ANSI_COLOR_RED "\x1b[31m"
#define ANSI_COLOR_GREEN "\x1b[32m"
#define ANSI_COLOR_YELLOW "\x1b[33m"
#define ANSI_COLOR_BLUE "\x1b[34m"
#define ANSI_COLOR_MAGENTA "\x1b[35m"
#define ANSI_COLOR_CYAN "\x1b[36m"
#define ANSI_COLOR_RESET "\x1b[0m"

int main (int argc, char const *argv[]) {

  printf(ANSI_COLOR_RED "This text is RED!" ANSI_COLOR_RESET "\n");
  printf(ANSI_COLOR_GREEN "This text is GREEN!" ANSI_COLOR_RESET "\n");
  printf(ANSI_COLOR_YELLOW "This text is YELLOW!" ANSI_COLOR_RESET "\n");
  printf(ANSI_COLOR_BLUE "This text is BLUE!" ANSI_COLOR_RESET "\n");
  printf(ANSI_COLOR_MAGENTA "This text is MAGENTA!" ANSI_COLOR_RESET "\n");
  printf(ANSI_COLOR_CYAN "This text is CYAN!" ANSI_COLOR_RESET "\n");

  return 0;
}
```

En sistemas Linux y Mac funciona de forma directa. En Windows es necesario instalar Windows Terminal y ejecutar el programa desde ahí.

3. Limpiar pantalla sin usar `system`

La función `system` (de la librería `stdlib`) permite ejecutar comandos del sistema operativo. Por ejemplo, con `system("cls")` se puede limpiar la pantalla en sistemas Windows. El problema es que el código no es portable (no se puede compilar ni ejecutar en otros sistemas operativos) y, por tanto, se debe evitar su uso. Una solución alternativa es el uso de los «ANSI escape codes» del punto anterior:

```
printf("\x1b[2J");
```

4. Introducir una pausa de una duración determinada

Ejemplo extraído de https://www.geeksforgeeks.org/time-delay-c/:

```c
// C function showing how to do time delay
#include <stdio.h>
// To use time library of C
#include <time.h>

void delay(int number_of_seconds)
{
	// Converting time into milli_seconds
	int milli_seconds = 1000 * number_of_seconds;

	// Storing start time
	clock_t start_time = clock();

	// looping till required time is not achieved
	while (clock() < start_time + milli_seconds);
}

// Driver code to test above function
int main()
{
	int i;
	for (i = 0; i < 10; i++) {
		// delay of one second
		delay(1);
		printf("%d seconds have passed\n", i + 1);
	}
	return 0;
}
```

5. Vector de cadenas de caracteres

Una cadena de caracteres es un vector de caracteres. Un vector de cadenas de caracteres (por ejemplo, una lista de textos) se puede pensar como una matriz de caracteres o como un puntero a cadenas de caracteres. Sin embargo, es más sencillo implementarlo mediante un vector de estructuras que tienen un único componente, una cadena de caracteres.

En el siguiente código se define una estructura que tiene un único componente, una cadena de caracteres. A continuación se define un vector de estructuras, que es, por tanto, un vector de cadenas de caracteres:

```c
#include <stdio.h>
#include <stdlib.h>
#define N 5 // Número de cadenas (longitud del vector)

typedef struct
{
  char texto[21];// Cada cadena tiene 20 caracteres max.
} cadena;

int main()
{
  cadena lista[N];
  int i = 0;
  //Lee los textos y los almacena en el vector
  for (i = 0; i < N; i++)
    scanf("%20s", lista[i].texto);
  //Muestra el resultado
  for (i = 0; i < N; i++)
     printf("%i: %s\n", i, lista[i].texto);

  return 0;
}
```