Sesión 5: *Hello C*

Programación para Sistemas

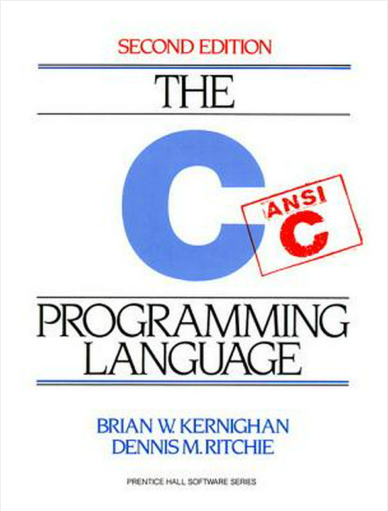
Ángel Herranz

Curso 2023-2024

Objetivos de la sesión

- Toma de contacto un poco más seria con C:
  - Estructura de un fichero
  - Ejemplo con varias funciones ademaś de `main`
  - Variables locales de tipo entero
- Practicar nuestro entorno de desarrollo como en el examen:
  - Conexión a `triqui`
  - Línea de comandos
  - Editar con editor de texto plano
  - Compilar
  - Ejecutar
  - Transferir ficheros a `triqui` si has trabajado en local

Consejo: Cómprate este libro



Hello C

```c
#include <stdio.h>

int main(int argc, char *argv[]) {
  printf("Hello C\n");
  return 0;
}
```

- Acceder a una máquina GNU/Linux (`triqui`)
- Crear el fichero `hello.c`
- Compilar con

  ```
  $ cc hello.c
  ```

- Ejecutar con

  ```
  $ ./a.out
  ```

CC: *The C Compiler*

- El compilador de C suele estar instalado y el nombre del programa suele ser **`cc`**
- Prueba: `which` **`cc`** y `man` **`cc`**
- Como cualquier programa **`cc`** admite argumentos, por ejemplo

```
cc -Wall -Werror -ansi -o hello hello.c
```

- El compilador *estándar* en GNU/Linux es GCC, el compilador de C de GNU
- El programa **`cc`** es, en general, el programa **`gcc`**
- En clase usaremos **`gcc`**

Fahrenheit-Celsius

Escribir un programa en C `fc.c` que imprima en la salida estándar una tabla de conversión de grados Fahrenheit a grados Celsius usando la fórmula

$$C = \frac{5}{9} \cdot (F - 32)$$

La tabla empieza en un mínimo (-20), llega hasta un máximo (240) y va dando saltos (de 20 en 20). Usa variables enteras para almacenar dichos datos. En la tabla usa tabuladores para separar los datos. Escribe una cabecera para indicar "F" y "C".

Fahrenheit-Celsius (minitutorial de C)

- Un fichero C es una colección de
  - *Includes* con declaraciones de *macros*, variables globales y funciones
  - Definiciones de variables globales (se pueden usar en cualquier función)
  - Definiciones de funciones (las funciones no se pueden anidar)
- Las funciones tiene un tipo de retorno, parámetros formales y cuerpo (como los métodos de Java pero en C no hay clases)
- Las declaraciones de variables son como en Java y los tipos *enteros* básicos son **`char`** (1 byte) e **`int`** (entre 2 y 8 bytes)
- La llamada a funciones es con sintaxis clásica: $f(x, y)$
- El control de flujo es similar al de Java: **`if`**, **`case`**, **`for`**, **`while`**, etc.

Fahrenheit-Celsius (estructura del programa)

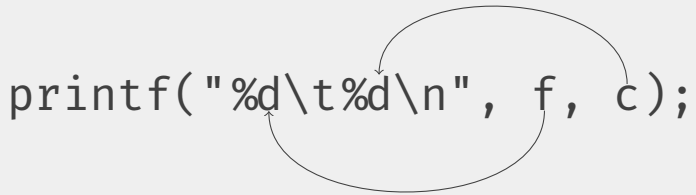
```
#include <stdio.h>

int fahrenheit_a_celsius(int f) {
  ⟨cuerpo de fahrenheit_a_celsius⟩
}

void imprimir_tabla(int min, int max, int paso) {
  ⟨cuerpo de imprimir_tabla⟩
}

int main() {
  ⟨cuerpo de main⟩
}
```

Fahrenheit-Celsius (main)

⟨*cuerpo de main*⟩ ≡

```
int min, max;
int paso = 20;
min = -20;
max = 240;
imprimir_tabla(min, max, paso);
return 0;
```

Fahrenheit-Celsius (printf)



- %d: conversión a decimal (se van sustituyendo en orden de aparición las conversiones que empiezan por % con los parámetros de `printf`)
- \t: caracter de tabulador
- \n: caracter de cambio de línea

Fahrenheit-Celsius (variación)

Modificar el programa `fc.c` para que los valores mínimo, máximo y paso de la tabla se pasen por argumentos, por ejemplo:

```
$ ./fc -20 240 20
```

El programa tiene que dar errores si el número de argumentos no es 3, si alguno de ellos no es un entero, si el mínimo es mayor que el máximo o si el paso es menor o igual que 0.

- Puedes empezar usando la función `atoi` (`man 3 atoi`)
- Como puedes ver la función `atoi` no detecta que un *string* no representa un entero, para resolver este problema puedes intentar usar la función `strtol` (`man 3 strtol`) hacer una función que compruebe que todos los caracteres de un *string* son dígitos (`man 3 isdigit`)