Lenguaje C

Lenguaje C

C es un lenguaje de programación:

- **estructurado**
- **compilado** de **bajo nivel**
- **tipado estático**

Entender estas propiedades nos ayudará a sacarle jugo a C y a entender el por qué de varias implementaciones

Paradigma estructurado

Un programa de C se construye a partir de los tipos primitivos atómicos, funciones, condicionales y bucles.

¿Y los objetos? ⟶ Structs

Compilado

Como es estructurado, en **tiempo de compilación** se verifica que los tipos tengan sentido y que no cambien en tiempo de ejecución. Gracias a esto, por ejemplo, sabe cuánta memoria reservar.

El compilador traduce todo el programa a código máquina (assembly) y, luego a binario. Esto hace que correr un programa hecho en C resulte **muy rápido.**

¿Binario? ¿Tenemos que aprender eso?

Compilado

¿Todas las computadoras entienden el mismo binario?

¿Es lo mismo una PC con intel que una con AMD?

¿Es lo mismo un programa compilado para una pc de hace 10 años que para una de ahora?

No a todo

Bajo Nivel

Siempre que hablemos de binario, assembly o estemos “más cerca” del hardware se dice que, estamos a más **“bajo nivel”**.

En C la interacción con el hardware es más **trabajosa** pero interactúa de manera más cercana al hardware, permiténdo esa personalización al programador.

En este curso, daremos mayor hincapié al **manejo de la memoria.**

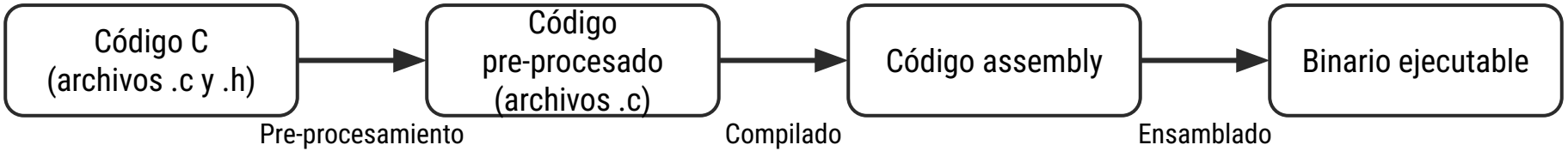
Tipado estático

Como vimos antes, la comprobación de tipos se realiza durante la compilación, y no durante la ejecución.

Hello World

Compilado en C

Compilado en C



El compilador que usaremos es GCC

Compilado en C

```
~ gcc -g -std=c99 -Wall -Wconversion
-Wno-sign-conversion -Werror hello-world.c -o
hello-world
```

El compilador que usaremos es GCC

Compilado en C

¿Es el mismo programa GCC el de la Mac Intel al de una Mac con M1?

JUSTIFIQUE SU RESPUESTA

Compilado en C

¿Qué pasa si cambio “hello, world!” por “hola, campeones del mundo?”

JUSTIFIQUE SU RESPUESTA

Sintaxis básica

Tipos de datos básicos

Tipo de dato	Descripción
void	Es el tipo “vacío”
char	Entero 2^8 combinaciones. ej.: [-128, 127]
int	Entero de al menos 2^16 combinaciones.
long	Entero de al menos 2^32 combinaciones.
float	Punto flotante, en general, de 32 bits
double	Punto flotante, en general,l el doble de bits que el tipo float
bool	Booleano [true, false]. **Hay que importarlo de stdbool.h**

Operadores aritméticos

Operador	Funcionamiento
`+`	Suma.
`–`	Resta.
`*`	Multiplicación.
`/`	División entera.
`%`	Resto de la división entera del primer número dividido el segundo.
`++`	Incrementa un número. Ejemplo: `a++;`
`––`	Decrementa un número.

Elementos de una función

Firma de una función

```
int add (int a, int b) {
    int total = a + b;
    return total;
}
```

Declaramos una variable

Casting o cambio de tipos

Para cambiar el tipo de una variable la sintaxis es:

```
(tipo_nuevo) variable
```

- Nos permite convertir de un tipo a otro siempre que sea posible.
- No todo casteo está libre de perder información.

Casting o cambio de tipos

Siempre que casteemos de un tipo más grande a uno más chico **podremos perder información**.

Operadores relacionales

Suponiendo A=1 y B=2

==	Verifica que sean iguales. A == B es false
!=	Verifica que sean distintos. A != B es true
>	Verifica si uno es mayor a otro. A > B es false
<	Verifica si uno es menor al otro.
>=	Verifica si uno es mayor o igual al otro.
<=	Verifica si uno es menor o igual al otro.

Operadores lógicos

Suponiendo A=true y B= false

& &	Es un AND. Ejemplos: A && B es false
| |	Es un OR. Ejemplos: A || B es true
!	Es un NOT. Ejemplos: !A es false !B es true A && !B es true

Bloque If/else

El bloque if o if else se puede escribir de varias formas, como bloque:

```
if (condición) {
   // código
}
```

Si el código es de una línea puede hacerse todo en una línea:

```
if (condición) //código;
```

Bloque If/else

Para el else el bloque es sencillamente:

```
if (condición) {
   // código condición verdadera
} else {
   // código condición falsa
}
```

Comentarios

Los comentarios de una línea se escriben empezando la línea con “//”.

```
// Documentando una sóla línea
```

Los comentarios en bloque se escriben empezando con /* y terminando con */.
Por ejemplo:

```
/*
* Verifica si un año es bisiesto.
* Si el número es negativo devuelve false.
*/
```

Comentarios

Es una práctica común y **necesaria** en la materia documentar las firmas de las funciones con lo que hacen y qué pasa en los casos borde.

Ejercicios

Implementamos dos funciones:

1. Una que determina si un número es par
2. Una que determina si un año es bisiesto

// Un año es bisiesto si es divisible por 4, pero no por 100, a menos que también sea divisible por 400

Bucle while

El while se escribe como:

```
while (condición) {
    // código mientras cumpla condición
}
```

O en una sola línea como:

```
while (condición) //código;
```

Bucle for

El for contiene 3 líneas de código a correr:

- Inicialización: Se corre antes de hacer cualquier otra cosa una sola vez
- Línea de test: Si es true, se continúa
- Línea de actualización: Luego de correr el código del for, si la línea de test es true se corre.

```
for (inicialización; test; actualización){

   // código

}
```

Bucle for

```
int fibonacci(int numero){
    if(numero==0) return 0;

    int anterior = 0;
    int actual = 1;
    int suma;

    for(int i=1; i<numero; i++){
        suma = anterior + actual;
        anterior = actual;
        actual = suma;
    }
    return actual;
}
```

¿Qué hace está función?

Ejercicio de la guía

Ejercicio 1.2 Escribir la función que dado n devuelve la suma de todos los números impares menores que n.