

# TRADUCTORES: COMPILADORES E INTERPRETES

Ana Magdalena Sotomayor

18 de febrero de 2015

## 1. INTRODUCCION

Los traductores son programas que nos permiten interactuar con los ordenadores. Son aquellos que toman como entrada un programa escrito en lenguaje simbolico y comprensible por los usuarios, que se denomina programa o codigo fuente y proporciona como salida otro programa escrito en un lenguaje comprensible por el hardware del ordenador, denominado programa objeto cuyo objetivo es que el ordenador realice el trabajo codificado.

Un compilador traduce completamente un programa fuente, escrito en un lenguaje de alto nivel, a un programa objeto, escrito en lenguaje ensamblador o maquina. El programa objeto resultante es independiente del compilador y puede ser procesado posteriormente sin volver a realizar la traduccion, y toda interaccion con el usuario estara controlada por el sistema operativo.

La traduccion por un compilador a la que se le llama compilacion, consta de dos etapas: la etapa de analisis del programa fuente y la etapa de sintesis del programa objeto. El analisis del texto fuente implica la realizacion de un analisis del lexico, de la sintaxis y de la semantica. La sintesis del programa objeto conduce a la generacion de codigo y su optimizacion. El compilador informa al usuario de cualquier error que se presente durante el analisis del texto y no crea el programa objeto hasta que los errores se hayan eliminado.

Por otro lado, el interprete permite que un programa fuente escrito en un determinado lenguaje vaya traduciendo y ejecutandose directamente, sentencia a sentencia, por el ordenador. El interprete capta una sentencia fuente, la analiza e interpreta, dando lugar a su ejecucion inmediata, no creandose, por tanto, un archivo o programa objeto almacenaje en memoria masiva para posteriores ejecuciones. La ejecucion del programa estara supervisada por el interprete.

## 2. TABLA COMPARATIVA PRINCIPALES TRADUCTORES

NOMBRE	PARADIGMA	CREADORES	APARICION	EXTENSIONES
C	Imperativo, Procedural, Estructurado	Dennis Ritchie	1972	.h .c
C++	Multiparadigma, orientado a objetos, imperativo, programacion generica.	Bjarne Stroustrup	1983	.h .hh .hpp .hxx .h++ .cc .cpp .cxx .c++
Fortran	Programacion de arreglos, programacion modular, orientada a objetos y generica.	John W. Backus	1957	.f .for .f90 .f95
Java	Multiparadigma, orientado a objetos, estructurado, imperativo, funional, generico, reflectivo, concurrente.	James Gosling	1955	.java .class .jar
Python	Multiparadigma, orientado a objetos, imperativo, reflexivo, funional.	Guido van Rossum	1991	.py -pyc .pyd .pyo .pyw
Ruby	Multiparadigma, orientado a objetos, imperativo, funcional, reflectivo.	Yukihiro Matsumoto	1995	.rb .rbw

## 3. EJEMPLOS DE CODIGO

### C

```
#include <iostream>

int main()
{
    printf("Hola! Tratar  de adivinar un n mero.\n");
    printf("Piensa un n mero entre 1 y 10.\n");
    sleep(5)
    printf("Ahora multiplicalo por 9.\n");
    sleep(5)
    printf("Si el n mero tiene 2 d gitos, s malos entre si: Ej. 36->3+6=9.
    Si tu n mero tiene un solo d gito, s male 0.\n");
    sleep(5)
    printf( "Al n mero resultante s male 4.\n");
    sleep(10)
    printf( "Muy bien. El resultrado es :)\n");
} \\ \hline
```

### 3.1. C++

```
#include <iostream>

int main()
{
    printf("Hola! TratarÃ© de adivinar un nÃºmero.\n");
    printf("Piensa un nÃºmero entre 1 y 10.\n");
    sleep(5)
    printf("Ahora multiplicalo por 9.\n");
    sleep(5)
    printf("Si el nÃºmero tiene 2 dÃ­gitos, sÃ©malos entre si: Ej. 36->3+6=9.
    Si tu nÃºmero tiene un solo dÃ­gito, sÃ©male 0.\n");
    sleep(5)
    printf( "Al nÃºmero resultante sÃ©male 4.\n");
    sleep(10)
    printf( "Muy bien. El resultado es :)\n");
}
```

### 3.2. Fortran

```
program Adivina

write(*,*) 'Hola! TratarÃ© de adivinar un nÃºmero.';
write(*,*) 'Piensa un nÃºmero entre 1 y 10.\n';
call sleep(5)
write(*,*) 'Ahora multiplicalo por 9.';
call sleep(5)
write(*,*) 'Si el nÃºmero tiene 2 dÃ­gitos, sÃ©malos entre si:
Ej. 36->3+6=9.
Si tu nÃºmero tiene un solo dÃ­gito, sÃ©male 0.';
call sleep(5)
write(*,*) 'Al nÃºmero resultante sÃ©male 4.';
call sleep(10)
write(*,*) 'Muy bien. El resultado es 13 :)';
end program Adivina
```

### 3.3. Java

```
public class Enjava {
    public static void main(String[] args) {

        System.out.println("Hola! TratarÃ© de adivinar un nÃºmero.");

        System.out.println("Piensa un nÃºmero entre 1 y 10.");
    }
}
```

```

try {
    Thread.sleep(1000);
} catch (InterruptedException e) {

    System.out.println("Ahora multiplicallo por 9.");
}
try {
    Thread.sleep(1000);
} catch (InterruptedException e)
{
    System.out.println("Si el número tiene 2 dígitos, súmalos entre si:
Ej. 36->3+6=9. Si tu número tiene un solo dígito, súmale 0.");
}
try {
    Thread.sleep(1000);
} catch (InterruptedException e)
{
    System.out.println("Al número resultante súmale 4.");
}
try {
    Thread.sleep(1000);
} catch (InterruptedException e)
{
    System.out.println("Muy bien. El resultado es 13. :)");
} }

}

```

### 3.4. Python

```

import time
print("Hola! Trataré de adivinar un número.")
print("Piensa un número entre 1 y 10.")
time.sleep(5)
print("Ahora multiplicallo por 9.")
time.sleep(5)
print("Si el número tiene 2 dígitos, sumalos entre si.
Si tu número tiene un solo dígito, sumale 0.")
time.sleep(5)
print( "Al número resultante sumale 4.")
time.sleep(10)
print( "Muy bien. El resultado es 13. :)") \\ \hline
\subsection{Ruby}
\begin{verbatim}
puts
    puts"Hola! Trataré de adivinar un número."

```

```
puts"Piensa un n mero entre 1 y 10."
sleep(5)
puts"Ahora multiplicalo por 9."
sleep(5)
puts"Si el n mero tiene 2 d gitos, s malos entre si:
Ej. 36->3+6=9. Si tu n mero tiene un solo d gito, s male 0."
sleep(5)
puts "Al n mero resultante s male 4."
sleep(10)
puts "Muy bien. El resultado es 13. :) "
```