Reporte sintético: Actividad 2

Ana Gabriela Carretas Talamante

13 de Febrero de 2015

1. Introducción

Los lenguajes de programación de alto nivel (que podemos leer como humanos) fueron diseñados para traducirse, ya sea como un lenguaje compilado o como un lenguaje interpretado. Para esto se usan los llamados **compiladores** e **interpretadores**.

Un compilador es un programa que permite traducir el código fuente de un programa en lenguaje de alto nivel, a otro lenguaje de nivel inferior. Analiza el programa y lo traduce al idioma "maquina". La acción fundamental los compiladores es equivalente a la de un traductor humano , que toma nota de lo que esta escuchando y reproduce por escrito en otra lengua.

Un interpretador es un programa informático capaz de analizar y ejecutar otros programas. Analiza el programa fuente y lo ejecuta directamente, o sea, en el ejemplo del traductor humano, éste sería uno que conforme a lo que está escuchando va ejecutando, sin generar ningún escrito, es decir que sobre la marcha va traduciendo.

Los lenguajes de programación que se introducen en esta actividad también pueden ser clasificados en compiladores e interpretadores.

Compiladores	**Interpretadores**
C, C++, Fortran, Java	Python, Ruby

A continuación se mostrará en la siguiente sección algunas características

de los seis lenguajes de programación mencionados en la tabla, incluyendo un ejemplo de código en el respectivo lenguaje.

2. Compiladores e Interpretadores Actuando

2.1. Tabla Comparativa

Nombre	Paradigma	Creadores	Año de aparición	Extensiones de archivo	Para Compilar
ANSI C	Imperativo (procedural), estructurado.	Dennis M. Ritchie	1972	.h .c	gcc
C++	Multiparadigma: orientado a objetos, imperativo, programación genérica.	Bjarne Stroustrup	1983	.h .hh .hpp .hxx .h++ .cc .cpp .cxx .c++	g++
Fortran90	Multiparadigma: estructurado, imperativo (procedural, orientado a objetos), genérico.	John Backus	1957	.f, .for, .f90, .f95	gfortran
Java	Orientado a objetos, imperativo.	Sun Microsystems (Oracle Corporation)	1995	.java, .class, .jar , .jad	javac
Python	Multiparadigma: orientado a objetos, imperativo, funcional, reflexivo.	Guido van Rossum	1991	.py, .pyc, .pyd, .pyo, .pyw	python
Ruby	Multiparadigma: orientado a objetos, reflexivo.	Yukihiro Matsumoto	1995	.rb, .rbw	ruby

2.2. "Juego: El adivinador"

- ANSI C

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

int main(void)
{
  puts("Hola! Tratare de adivinar un numero.");
  puts("Piensa en un numero del 1 al 10.");
  sleep( 5 );
  puts("Ahora multiplicalo por 9.");
  sleep( 5 );
  puts("Si el numero tiene dos digitos, sumalos entre si. Ej. 36 -> 3+6=9.
  sleep( 5 );
  puts("Al numero restante sumale 4.");
  sleep( 10 );
  puts("Muy bien. EL resultado es 13 :)");
  return EXIT_SUCCESS;
}
```

- C++

```
#include <iostream>
#include <unistd.h>

int main()
{
    std::cout << "Hola! Trataré de adivinar un número. Piensa en un número
sleep(5);
std::cout << "Ahora multiplícalo por 9.\n";
sleep(5);
std::cout << "Si el número tiene 2 dígitos, súmalos entre si: Ej. 36 -> 3+
sleep(5);
```

```
std::cout << "Al número resultante súmale 4.\n";
sleep(10);
std::cout << "Muy bien. El resultado es 13 :)\n";
return(0);
}
```

- Fortran 90

```
PROGRAM Adivinador
     WRITE(*,*)  "Hola! Trataré de adivinar un número. Piensa en un número
call sleep (5)
WRITE(*,*)  "Ahora multiplicalo por 9."
call sleep (5)
WRITE(*,*)  "Si el número tiene 2 dígitos, súmalos entre si: Ej. 36 -> 3+6
call sleep (5)
WRITE(*,*)  "Al numero resultante sumale 4."
call sleep (10)
WRITE(*,*)  "Muy bien. EL resultado es 13 :)"
END PROGRAM
```

- Java

```
class Adivinador {
  static public void main( String args[] ) {
      System.out.println( "Hola! Trataré de adivinar un número. Piensa en
try {
    Thread.sleep(5000);
} catch(InterruptedException ex) {
    Thread.currentThread().interrupt();
}
System.out.println( "Ahora multiplicalo por 9." );
try {
    Thread.sleep(5000);
} catch(InterruptedException ex) {
    Thread.currentThread().interrupt();
}
System.out.println( "Si el número tiene 2 dígitos, súmalos entre si: Ej. 3
try {
```

```
    Thread.sleep(5000);
} catch(InterruptedException ex) {
    Thread.currentThread().interrupt();
}
System.out.println( "Al numero resultante sumale 4." );
try {
    Thread.sleep(10000);
} catch(InterruptedException ex) {
    Thread.currentThread().interrupt();
}
System.out.println( "Muy bien. EL resultado es 13 :)" );
  }
}
```

- Python

```
import time
print "Hola! Trataré de adivinar un número. Piensa un número entre 1 y 10.
time.sleep(5)
print "Ahora multiplícalo por 9."
time.sleep(5)
print "Si el número tiene 2 dígitos, súmalos entre sí. Ej. 36 -> 3+6=9.  S
time.sleep(5)
print "Al número resultante súmale 4."
time.sleep(10)
print "Muy bien. El resultado es 13 :)"
```

- Ruby

```
puts "Hola! Trataré de adivinar un número."
puts "Piensa un número entre 1 y 10."
sleep(5)
puts "Ahora multiplícalo por 9."
sleep(5)
puts "Si el número tiene 2 dígitos, súmalos entre si: Ej. 36 -> 3+6=9. Si
sleep(5)
puts "Al número resultante súmale 4."
sleep(10)
puts "Muy bien. El resultado es 13 :) "
```