Reporte Sintético

Carlos Medina

12-2-15

El siguiente reporte tratará de comparar, resaltar, diferenciar y ejemplificar varios compiladores e intrerpretadores que se necesitan y se piden en la actividad 2 (2015-1). Un compilador es un programa informático que traduce un programa escrito en un lenguaje de programación a otro lenguaje de programación, generando un programa equivalente que la máquina será capaz de interpretar. 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 (típicamente lenguaje de máquina). Un intérprete o interpretador es un programa informático capaz de analizar y ejecutar otros programas, escritos en un lenguaje de alto nivel. Los intérpretes se diferencian de los compiladores en que mientras estos traducen un programa desde su descripción en un lenguaje de programación al código de máquina del sistema, los intérpretes sólo realizan la traducción a medida que sea necesaria, típicamente, instrucción por instrucción, y normalmente no guardan el resultado de dicha traducción. Los compiladores e interpretadores que usaremos en esta ocasión son:

- `ANSI C`
- `C++`
- `Fortran 90`
- `Java`
- `Python`
- `Ruby`

A continucación le mostraremos una tabla comparativa de los lenguajes arriba mencionados.

Tabla comparativa					
Nombre	Paradigma	creadores	año de aparición	extensiones de archivo	C o ta ar fu
ANSI C	Imperativo (Procedural), Estructurado	Dennis Ritchie, Ken Thompson	1972	.c, .h	g
C++	multiparadigma: orientado a objetos, imperativo, programación genérica.	Bjarne Stroustrup	1983	.h, .hh, .hpp, .hxx, .h++, .cc, .cpp, .cxx, .c++	g
Fortran 90	multi-paradigm: structured, imperativo (procedural, orientado a objetos), generico	John Backus	1957	.f, .for, .f90, .f95	g
Java	orientado a objetos, structured, imperativo, funcional, generico, reflectivo, concurrente	Sun Microsystems (Oracle Corporation)	1995	.java, .class, .jar, .jad	ja
Python	multiparadigma: orientado a objetos, imperativo, funcional, reflexivo	1991	Guido van Rossum	.py, .pyc, .pyd, .pyo, .pyw	p
Ruby	multiparadigma: orientado a objetos, reflexivo	Yukihiro Matsumoto	1995	.rb, .rbw	

Ahora, veremos un ejemplo de compilación/interpretación con un programa Adivina la mente en cada uno de los lenguajes.

ANSI C:

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


int main(void)
{
  puts("Hola! Trataré de adivinar un número."
"Piensa un número entre 1 y 10.Hello World!");
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 tu nú
sleep(5);
puts("Al número resultante súmale 4.");
sleep(10);
puts("Muy bien. El resultado es 13 :3");
return(EXIT_SUCCESS);
}
```

C++

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

int main()
{
    std::cout << "Hola! Trataré de adivinar un número. Piensa en un número entr
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+6=9.
sleep(5);
std::cout << "Al número resultante súmale 4.\n";
sleep(10);
```

```
std::cout << "Muy bien. El resultado es 13 :3\n";
return(0);
}
```

FORTRAN 90

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

Java

```
class HelloWorld {
  static public void main( String args[] ) {
 System.out.println( "Hola! Trataré de adivinar un número. Piensa un número ent
try {
    Thread.sleep(5000);
} catch(InterruptedException ex) {
    Thread.currentThread().interrupt();
}
System.out.println( "Ahora multiplícalo 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. 36 ->
try {
```

```
    Thread.sleep(5000);
} catch(InterruptedException ex) {
    Thread.currentThread().interrupt();
}
System.out.println( "Al número resultante súmale 4." );
try {
    Thread.sleep(10000);
} catch(InterruptedException ex) {
    Thread.currentThread().interrupt();
}
System.out.println( "Muy bien. El resultado es 13 :3 " );
}

}
```

Python

```
 import time
while True:
  print "Hola! Tratare de adivinar un numero. Piensa un numero entre 1 y 10."
time.sleep(5)
import time
while True:
  print "Ahora multiplicalo por 9."
time.sleep(5)
import time
while True:
  print "Si el numero tiene 2 digitos, sumalos entre si: Ej. 36 -> 3+6=9. Si tu
time.sleep(5)
import time
while True:
  print "Al numero resultante sumale 4."
time.sleep(10)
import time
while True:
  print "Muy bien. El resultado es 13 :3 "
```

Ruby

```
 puts "Hola! Tratare de adivinar un numero. Piensa un numero entre 1 y 10."
sleep(5)
puts "Ahora multiplicalo por 9."
sleep(5)
puts "Si el numero tiene 2 digitos, sumalos entre si: Ej. 36 -> 3+6=9. Si tu nu
sleep(5)
puts "Al numero resultante sumale 4."
sleep(10)
puts "Muy bien. El resultado es 13 :3 "
```