Índice

1. A02. Lenguajes de programación y herramientas de desarrollo

1.1 Introducción

En la actividad que nos ocupa se pretenden los siguientes objetivos:

- Diferenciar lenguajes informáticos, clasificarlas e identificar y caracterizar los lenguajes de programación más populares.
- Reconocer las características, código generado y herramientas utilizadas en la edición, compilación, enlace y ejecución para lenguajes de programación compiladas, interpretadas, de máquina virtual o de ejecución administrada.

1.2 Actividad

Clasificación de los lenguajes informáticos

Un lenguaje informático es un lenguaje que permite comunicarse con el ordenador y está formada por un conjunto de símbolos y palabras que siguen unas normas de sintaxis.

No existe una clasificación de los lenguajes informáticos adoptada por la mayoría de los autores si no que hay grandes diferencias entre ellos. Una clasificación posible es: lenguajes de marcas, especificación, consulta, transformación y programación.

Lenguajes de marcas

Permiten colocar distintivos o señales en el texto que serán interpretadas por aplicaciones o procesos. Ejemplos, de lenguajes de marcas:

- El lenguaje XML (*extensible Markup Language*) que es un metalenguaje extensible pensada para la transmisión de información estructurada y que puede ser validada.
- Los lenguajes HTML (*Hiper Text Markup Language*) o XHTML (*extensible Hiper Text Markup Language*) =XML+HTML que sirven ambas para publicar hipertexto en la Word Wide Web.

Ejemplo de código XML editado en NetBeans:

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Ejemplo básico de XML
-->
<alumnos>
    <alumno>
        <nombre>Pepe</nombre>
        <apellidos>Ruíz Arias</apellidos>
    </alumno>
    <alumno>
        <nombre>María Dolores</nombre>
        <apellidos>González Paz</apellidos>
    </alumno>
</alumnos>
```

Ejemplo de código XHTML editado en Notepad++:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<!-- Ejemplo muy básico de XHTML con estilo externo -->
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Ejemplo básico de xhtml</title>
        <meta   content="text/html; charset=utf-8"
                http-equiv="Content-Type" />
        <link   rel="stylesheet" href="ejemplo_css.css"
                type="text/css" />
    </head>
    <body>
        <h1>Cabecera principal</h1>
        <p class="principal">Este párrafo contiene texto y un enlace
la
            <a href ="http://www.realacademiagalega.org/">Real
Academia Gallega
            </a>.
        </p>
    < /body>
</html>
```

Lenguajes de especificación

Describen algo de forma precisa. Por ejemplo CSS (*Cascading Style Sheets*) es un lenguaje formal que especifica la presentación o estilo de un documento HTML, XML o XHTML. Ejemplo de código CSS que se podría aplicar al ejemplo XHTML anterior y editado en Notepad++:

```
body{
    font-family: "MS Sans Serif", Geneva, sans-serif;
    font-size: 15px;
    color: Black;
    border: black 2px double;
    padding: 40px;
    margin: 20px;}
h1 {
    font: 40px "Times New Roman", Times, serif;
    font-weight: bolder;
    word-spacing: 25px;}
p.principal{
    text-align: center;
    font: 10px "MS Serif", "New York", serif;}
```

Lenguajes de consultas

Permiten sacar o manipular información de un grupo de información. Por ejemplo, el lenguaje de consultas SQL (*Standard Query Language*) permite buscar y manipular información en bases de datos relacionais y el lenguaje XQuery permite buscar y manipular información en bases de datos XML nativas. Ejemplo de script SQL:

```
CREATE DATABASE  `empresa` ;

USE `empresa`;

CREATE TABLE `centros` (
  `cien_num` int(11) NOT NULL default '0',
  `cien_nom` char(30) default NULL,
  `cien_dir` char(30) default NULL,
  UNIQUE KEY `numcen` (`cien_num`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

INSERT INTO `centros` VALUES
```

```
        (10,'SED CENTRAL','C/ ALCALA, 820-MADRID'),
        (20,'RELACION CON CLIENTES','C/ ATOCHA, 405-MADRID');
 CREATE TABLE `deptos` (
  `dep_num` int(11) NOT NULL default '0',
  `dep_cien` int(11) NOT NULL default '0',
  `dep_dire` int(11) NOT NULL default '0',
  `dep_tipodir` char(1) default NULL,
  `dep_presu` decimal(9,2) default NULL,
  `dep_depen` int(11) default NULL,
  `dep_nom` char(20) default NULL,
  UNIQUE KEY `numdep` (`dep_num`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;

INSERT INTO `deptos` VALUES
        (122,10,350,'F','60000.00',120,'PROCESO DE DATOS'),
        (121,10,110,'P','200000.00',120,'PERSONAL'),
        (120,10,150,'P','30000.00',100,'ORGANIZACION'),
        (112,20,270,'F','90000.00',110,'SECTOR SERVICIOS'),
        (111,20,400,'P','111000.00',110,'SECTOR INDUSTRIAL'),
        (110,20,180,'P','15000.00',100,'DIRECCION COMERCIAL'),
        (130,10,310,'P','20000.00',100,'FINANZAS'),
        (200,20,600,'F','80000.00',100,'TRANSPORTES'),
        (100,10,260,'P','120000.00',NULL,'DIRECCION GENERAL');
```

Ejemplo de expresión FLOWR *(for, let, order by, where, return*) en XQuery:

```
for $libro in doc("libros.xml")/bib/libro
let $editorial := $libro/editorial
where $editorial="MCGRAW/HILL" or contains($editorial, "Toxelsotos")
return $libro
```

Lenguajes de transformación

Actúan sobre una información inicial par obtener otra nueva. Por ejemplo, el lenguaje XSLT (*extensible Stylesheet Language Tranformations*) permite describir las transformaciones que se van a realizar sobre un documento XML para obtener otro archivo. Ejemplo de transformación XSL sencilla editada en NetBeans, que cuando actúa sobre el ejemplo XML anterior obtiene una página HTML con una lista de los alumnos:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Ejemplo sencillo de transformación XSL -->
<xsl:stylesheet xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
                version="1.0">
    <xsl:output method="html"/>
    <xsl:template match="alumnos">
        <html>
            <head>
                <title>fundamentos.xsl</title>
            </head>
            <body>
                <h1>Alumnos</h1>
                <ul>
                    <xsl:for-each select="alumno">
                        <li>
                            <xsl:value-of select="apellidos"/>,
                            <xsl:value-of select="nombre"/>
                        </li>
                    </xsl:for-each>
                </ul>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Lenguajes de programación

Permiten comunicarse con los dispositivos hardware y así poder realizar un determinado proceso y para eso pueden manejar estructuras de datos almacenados en memoria interna o y externa, y utilizar estructuras de control. Disponen de un léxico, y tienen que cumplir reglas sintácticas y semánticas.

El léxico es el conjunto de símbolos que se pueden usar y pueden ser: identificadores (nombres de variables, tipos de datos, nombres de métodos, etcétera), constantes, variables, operadores, instrucciones y comentarios.

Las reglas de sintaxis especifican la secuencia de símbolos que forman una frase bien escrita en ese lenguaje, es decir, para que no tenga faltas de ortografía.

Las reglas de semántica definen como tienen que ser las construcciones sintácticas y las expresiones y tipos de datos utilizadas.

Ejemplo de código Java sencillo editado en NetBeans:

```
package parimpar;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        short n;
        Scanner teclado = new Scanner(System.in);
        System.out.printf("Teclee el número entero entre %d y %d:",

             Short.MÍ_VALUE, Short.MAX_VALUE);
        n = teclado.nextShort();
        if(n%2==0){
            System.out.printf("%d es par\n",n);
        }
        else{
            System.out.printf("%d es impar\n",n);
        }
    }
}
```

Clasificación de los lenguajes de programación

Los lenguajes de programación pueden clasificarse según lo lejanas o próximas que están del hardware, por generaciones, por el paradigma de la programación, por la forma de traducirse el lenguaje máquina y de ejecutarse, y en la arquitectura cliente servidor.

Clasificación según la distancia al hardware

Pueden clasificarse en lenguajes de bajo y alto nivel.

Lenguajes de bajo nivel

Están basados directamente en los circuitos electrónicos de la máquina por lo que un programa escrito para una máquina no podrá ser utilizada en otra diferente. Pueden ser lenguaje máquina o lenguaje ensambladora.

El lenguaje máquina es código binario (ceros y unos) o hexadecimal (números de 0 a 9 y letras de La a F) que actúa directamente sobre el hardware. ES el único lenguaje que no necesita traducción ya que el procesador reconoce las instrucciones directamente.

La codificación en lenguaje ensambladora es mnemotécnica, es decir, utiliza etiquetas para describir ciertas operaciones. ES necesario traducir el código ensamblador el lenguaje máquina para que el procesador reconozca las instrucciones.

Tarea 1. Buscar en internet un trozo de código escrito en lenguaje ensamblador.

Lenguajes de alto nivel

Intentan acercarse al lenguaje humano y se separan del conocimiento interno de la máquina y por eso necesitan traducirse el lenguaje máquina. Esta traducción hace que la ejecución sea más lenta que en los lenguajes de bajo nivel, pero al no depender del procesador pueden ejecutarse en diferentes ordenadores.

Clasificación por generaciones

Pueden clasificarse en 5 generaciones.

- Primera generación (1GL) formada por los lenguajes de programación utilizados en los primeros ordenadores: lenguaje máquina y ensamblador.
- Segunda generación (2GL) formada por el lenguaje macroensamblador que es el lenguaje ensamblador combinado con instrucciones de control y de manejo de datos más complejos. Estos lenguajes son específicos para una familia de procesadores y el hardware relacionado con el. Aún se sigue utilizando para programar los núcleos (*kernel*) de los sistemas operativos y los controladores de algunos dispositivos (*device drivers*).
- Tercera generación (3GL) formada por la mayor parte de los lenguajes de alto nivel actuales. El código es independiente de la máquina y el lenguaje de programación es parecido al lenguaje humano. Por ejemplo, Java, C, C++, PHP, JavaScript, y Visual Basic.
- Cuarta generación (4GL) formada por lenguajes y entornos diseñados para una tarea o propósito muy específico como acceso a bases de datos, generación de informes, genreación de interfaces de usuario, etcétera. Por ejemplo, SQL, Informix 4GL, y Progress 4GL.
- Quinta generación (5GL) formada por lenguajes en las que el programador establece el problema a resolver y las condiciones a cumplir. Se usan en inteligencia artificial, sistemas basados en el conocimiento, sistemas expertos, mecanismos de inferencia o procesamiento del lenguaje natural. Por ejemplo, Prolog, Smalltalk y Lisp.

Clasificación por el paradigma de la programación

Un paradigma de programación es una metodología o filosofía de programación a seguir con un núcleo central incuestionable, es decir, los lenguajes que utilizan el mismo paradigma de programación utilizarán los mismos conceptos básicos para programar. La evolución de las metodologías de programación y la de los lenguajes van parejas a lo largo del tiempo. Pueden clasificarse en dos grandes grupos: el grupo de los que siguen el paradigma imperativo y el de los que siguen el paradigma declarativo.

Paradigma imperativo

El código está formado por una serie de pasos o instrucciones para realizar una tarea organizando o cambiando valores en memoria. Las instrucciones se ejecutan de forma secuencial, es decir, hasta que no se ejecuta una no se pasa a ejecutar la siguiente. Por ejemplo, Java, C, C++, PHP, JavaScript, y Visual Basic.

Dentro de los lenguajes imperativos se distingue entre que las que siguen la metodología estructurada y las que siguen la metodología orientada a objetos.

A finales de los años 60 nació la metodología estructurada que permitía tres tipos de estructuras en el código: la secuencial, la alternativa (basada en una decisión) y la repetitiva (bucles). Los programas están formados por datos y esas estructuras.

La metodología estructurada evolucionó añadiendo el módulo como componente básico dando lugar a la programación estructurada y modular. Los programas estaban formados por módulos o procesos por un lado y datos por otro. Los módulos necesitaban de unos datos de entrada y obtenían otros de salida que a su vez podían ser utilizados por otros módulos. Por ejemplo, C es un lenguaje estructurado y modular.

En los años 80 apareció la metodología orientada a objetos que considera el objeto como elemento básico. Esta metodología se popularizo a principios de los 90 y actualmente es la más utilizada. Por ejemplo, C++ y Java son lenguajes orientados a objetos. Cada objeto sigue el patrón especificado en una clase. Los programas contienen objetos que se relacionan o colaboran con otros objetos de su misma clase o de otras clases. La clase está compuesta de atributos (datos) y métodos (procesos) y puede tener las propiedades:

- Herencia. Pueden declararse clases hija que heredan las propiedades y métodos de la clase madre. Los métodos heredados pueden sobrecargarse, es decir, dentro de la misma clase puede aparecer definido con distinto comportamiento el mismo método con diferente firma (número de parámetros y tipo de los mismos).
- Polimorfismo. Propiedad que permite que un método se comporte de diferente manera dependiendo del objeto sobre lo que se aplica.
- Encapsulamiento. Permite ocultar detalles internos de una clase.

Paradigma declarativo.

El código indica que es lo que se quiere obtener y no como se tiene que obtener, es decir, es un conjunto de condiciones, proposiciones, afirmaciones, restricciones, ecuaciones o transformaciones que describen el problema y detallan su solución. El programador tiene que pensar en la lógica del algoritmo y no en la secuencia de órdenes para que se lleve a cabo. Por ejemplo, Lisp y Prolog son lenguajes declarativos.

Clasificación por la forma de traducirse el lenguaje máquina y ejecutarse

Se llama código fuente al código escrito en un lenguaje de programación simbólico mediante una herramienta de edición. Este código se guarda en un archivo conocido como archivo fuente y tendrá que traducirse el lenguaje máquina para poder ejecutarse. La traducción puede hacerse mediante compiladores y/o intérpretes.

Puede considerarse que hay tres grupos de lenguajes de programación habida cuenta la forma de traducirse el lenguaje máquina y de ejecutarse: Lenguajes compilados, interpretados, y de máquina virtual o ejecución administrada. Algunos lenguajes como por ejemplo Prolog pueden ser considerados como compilados o interpretados ya que disponen de compiladores o intérpretes.

Lenguaje compilado

Estos lenguajes disponen de un compilador o programa que traduce el código fuente a código máquina creando el archivo ejecutable en tiempo de compilación. En tiempo de ejecución puede lanzarse el archivo ejecutable en la plataforma para la que sirve el compilador. Algunas características de este tipo de lenguajes son:

- Existe un archivo ejecutable para la máquina real.
- El proceso de traducción se hace en tiempo de compilación y una sola vez.
- La ejecución es muy rápida ya que el ejecutable está en lenguaje máquina.
- El ejecutable sólo funciona en la plataforma para la que fue creado.

- El usuario que ejecuta no tiene que conocer el código fuente, sólo tiene el código ejecutable que no es manipulable fácilmente para obtener el código fuente, y como consecuencia el programador tiene el código fuente más protegido.
- El ejecutable no se generará se existen errores léxicos, sintácticos o semánticos.
- Interrumpir la ejecución puede ser difícil para el sistema operativo y puede afectar a la plataforma.
- La modificación del código fuente implica volver a generar el archivo ejecutable.

Lenguaje interpretado

Estos lenguajes precisan de un intérprete o programa en memoria para que en tiempo de ejecución se vaya traduciendo el código fuente a lenguaje máquina. Algunas características de este tipo de lenguajes son:

- No existe un archivo ejecutable.
- El proceso de traducción se hace cada vez que se ejecuta.
- La ejecución es lenta ya que al proceso de ejecución tiene que sumarse el de traducción.
- El archivo puede ejecutarse en diferentes plataformas siempre que exista intérprete para ellas.
- El usuario utiliza el código fuente para ejecutar el programa.
- Los errores de tipo léxicos, sintácticos o semánticos pueden aparecer en la ejecución.
- Interrumpir la ejecución sólo afecta normalmente al intérprete y no a la plataforma.
- Puede ejecutarse en cualquier plataforma siempre que haya intérprete para ella.
- La modificación del código fuente no requiere ninguna operación extra antes de ejecutar el programa.

Lenguaje de máquina virtual o de ejecución administrada

Estos lenguajes precisan inicialmente de un compilador que en tiempo de programación traducen el código fuente a un código intermedio multiplataforma, y a continuación necesita de otro software que traduzca ese código intermedio a código máquina. En el caso de lenguajes de máquina virtual como Java ocurre que:

- En tiempo de compilación el compilador Java genera el código intermedio llamado bytecode Java independiente de la plataforma.
- En tiempo de ejecución, necesita de una máquina virtual Java (JVM) que interprete el bytecode. Esta máquina virtual es un software que simula una máquina real con su propio sistema operativo y que hace de intermediaria con la máquina real, por lo que las máquinas virtuales son diferentes para Linux, Windows, Mac y Solaris.

En el caso de lenguajes de ejecución administrada cómo las de la plataforma .NET[1] de Microsoft ocurre que:

- En tiempo de compilación, el compilador genera el código CIL (*Common Intermediate* Language) independiente de la plataforma.

[1] Esta plataforma es un conjunto de tecnologías de software que tiene un contorno de trabajo (.NET Framy work) y varios lenguajes de programación como por ejemplo VisualBasic.NET, C# o C++. El contorno de trabajo, incluido en el sistema operativo Windows, incluye un contorno de ejecución administrado llamado CRL (*Common Runtime Language*) que ejecuta código y proporciona servicios que facilitan el proceso de desarrollo.

- Para ejecutar es necesario tener en el equipo cliente a versión de .NET Framework (y por tanto de CRL) adecuada y entonces se hace la última fase de la compilación, en la que el CRL tra duciráel código intermedio a código máquina mediante uno compilador JIT (*Just InT ime*).

Algunas características de este tipo de los lenguajes de máquina virtual o de ejecución administrada son:

- Existe un código intermedio que se ejecuta en un software específico, pero no es un ejecutable.
- El proceso de traducción inicial se hace antes de la ejecución y la traducción final se hace cada vez que se ejecuta.
- La ejecución no es tan rápida cómo en los lenguajes compilados, pero es más rápido que en los lenguajes interpretados.
- El archivo puede ejecutarse en diferentes plataformas siempre que exista el software y específico para ello.
- El usuario no tiene el código fuente, sólo el código intermedio que no es manipulable fácilmente para obtener el código fuente por lo que el programador tiene el código fuente más protegido.
- Los errores de tipo léxico, sintáctico o semántico se detectan en la fase de compilación, y se existen no se generará el código intermedio.
- Interrumpir la ejecución sólo afecta normalmente al intérprete y no a la plataforma.
- La modificación del código fuente implica volver a repetir el proceso de compilación.

Clasificación en la arquitectura cliente-servidor

La arquitectura cliente-servidor consiste básicamente en un programa cliente que realiza peticiones a un servidor. Esta arquitectura es más útil cuando el cliente y el servidor están comunicados mediante una red, aunque también se puede aplicar cuándo están en la misma maquina. En esta arquitectura se diferencia entre lenguajes que se ejecutan:

- del lado del servidor como PHP.
- del lado del cliente como JavaScript.

Para explicar esto se puede suponer un navegador cliente y un servidor web con intérprete de PHP y la situación mostrada en los siguientes pasos:

- Un usuario desde uno navegador cliente pide a un servidor Web una página HTML que tiene incrustado código PHP y código JavaScript.
- El servidor Web procesa la petición, interpreta el código PHP, lo ejecuta colocando el resultado de la ejecución en el sitio donde estaba el código PHP y devuelve al cliente una página web con código HTML y JavaScript. El servidor Web debería de ejecutar las ordenes relativas a la manejo de una base de datos en un servidor de base de datos si el código PHP había tenido ese tipo de instrucciones.
- El navegador cliente interpreta el código JavaScript y muestra la página al usuario.

El usuario puede interactuar con la página ocurriendo que cada vez que hace peticiones al servidor se recarga la página completamente con la respuesta del servidor.

Existe una técnica denominada AJAX (*Asynchronous JavaScript And XML*), que permite que JavaScript procese algún evento iniciado por el usuario haciendo una petición al servido que este responde con el resultado en XML. JavaScript procesa el resultado XML actualizando secciones de la página sin tener que recargarla totalmente y logrando así una

interacción asíncrona entre servidor y cliente.

Lenguajes más difundidos

Para saber cuáles son los lenguajes más difundidos pueden consultarse los siguientes índices o clasificaciones:

- Índice Tiobe (http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html) que basa la clasificación de los lenguajes de programación en el número de ingenieros cualificados en cada lenguaje en todo el mundo, cursos de lenguajes ofertados, proveedores que trabajan sobre los lenguajes, buscas realizadas en Google, Bing, Yahoo, Wikipedia, Amazon, Youtube y Baidu y una serie de premisas como por ejemplo que el lenguaje exista como lenguaje de programación en Wikipedia y que tenga por lo menos 10000 visitas en Google.
- Índice PYPL o PopularitY *of Programming Language index* (https://sites.google.com/site/pydatalog/pypl/PyPL-PopularitY-of-Programming-Language) que basa la clasificación de los lenguajes de programación en el análisis de la frecuencia de búsqueda de tutoriales o guías de los lenguajes de programación en Google utilizando Google Trends.
- La clasificación Redmonk (http://redmonk.com/sogrady/2014/01/22/language-ránkings-1-14/) que no basa la clasificación nos buscadores sino en los proyectos albergados en el repositorio GitHub y en las preguntas de la web de StackOverflow orientada a programadores. Incluye lenguajes informáticos y no sólo lenguajes de programación.
- La clasificación Trendyskills (http://trendyskills.com/) se basa en las ofertas de empleo para los lenguajes de programación en España, USA, UK, Alemania, Suecia, Países Bajos, Irlanda, Suiza, Austria, Bélgica, Finlandia, República Checa y Grecia e incluye lenguajes informáticos y no sólo lenguajes de programación.

Los lenguajes comunes a todos estos índices en el año 2014 en los 10 primeros puestos son: C, C++, Java, C#, PHP, Python y JavaScript.

10 Lenguajes de programación más populares en 2014

Índice Tiobe	Índice PYPL	Clasificación Redmonk	Clasificación Trendyskills
C	Java	JavaScript	Java
Java	PHP	Java	JavaScript
Objective-C	Python	PHP	C#
C++	C#	C#	HTML
Visual Basic	C++	Python	PHP
C#	C	C++	HTML5
PHP	JavaScript	Ruby	XML
Python	Objective-C	C	C++
JavaScript	Ruby	Objective-C	C
Transact-SQL	Visual Basic	CSS	Python

C

C es un lenguaje creada por Dennis Ritchie en 1972 en los laboratorios Bell para codificar el sistema operativo UNIX. Está catalogada como un lenguaje de alto nivel, estructurada y modular, pero con muchas características de bajo nivel como por ejemplo utilizar punteros para hacer referencia a una posición física de la memoria RAM. Estas características posibilitan trabajar muy cerca de la máquina, pero los programas son más complicados y

propensos a tener más errores. Influyó en el diseño de los lenguajes C++, C#, Java, PHP, JavaScript entre otros. Ejemplo de código C:

```c
/*
 * Programa C que suma los números enteros del 1 al 20
 */
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char** argv) {
    int i;              /* contador de números enteros */
    long int suma;      /* sumador de los números enteros */
    int final;
    suma = 0;
    for (i = 1; i <=20; i++) {
        suma = suma + i;
    }
    printf("Programa C\en la suma total y:  %ld\n", suma);
    printf("\nTeclee cualquiera numero para finalizar...");
    scanf("%d",&final);
    fflush(stdin);
    return (EXIT_SUCCESS);
}
```

C++

Diseñado en 1980 por Bjarne Stroustrup para extender C con mecanismos que permitan a POO (programación orientada a objetos). Ejemplo de código C++ editado en NetBeans:

```cpp
/*
 * Programa C++ que suma los números enteros del 1 al 20
 */
#include <iostream>
int main(int argc, char**argv) {
    int i;              /* contador de números enteros */
    long int suma;      /* sumador de los números enteros */
    suma = 0;
    for (i = 1; i <=20; i++) {
        suma = suma + i;
    }
    std::cout <<"La suma total y "<< suma<< std::endl;
    return 0;
}
```

Java

Es un lenguaje de programación orientado a objetos desarrollada por Sun Microsystems en 1995. Toma mucha de su sintaxis de C y C++ pero con un modelo de objetos más simple y elimina las herramientas de bajo nivel que se utilizaban en C. Ejemplo de código Java editado en NetBeans:

```java
package sumaenteiros;
/*
 * File: Main.java
 * Author: profesor
 * Date: 14/08/2011 12:25:00
 * Objetivo: visualizar la suma de los 20 primeros números naturales
 */
public class Main {

    public static void main(String[] args) {
        int suma=0;     /* sumador de los números enteros */
        for (int i=1;i<=20;i++)
        {
            suma=suma+i;
```

```
        }
        System.out.printf("Ejemplo Java\n");
        System.out.printf("Suma de los 20 primeros números naturales
= %d\n",suma);
    }
}
```

C#

Es un lenguaje de programación orientado a objetos desarrollado por Microsoft en el año 2000 como parte de la plataforma .NET. Visual C# proporciona un editor de código avanzado, diseñadores de interface de usuario, y numerosas herramientas para facilitar lo desarrollo de aplicaciones en C# y .NET Framework. Ejemplo de código C# creado con Visual Studio 2013 y editado con Notepad++:

```
using System;

namespace ejemplo_csharp_consola
{
    class Program
    {
        static void Main(string[] args)
        {
            Consuele.WriteLine("Hola Mundo");
        }
    }
}
```

PHP

PHP *(Hypertext PreProcessor)* es un lenguaje interpretado del lado del servidor que se puede usar para crear cualquier tipo de programa pero que donde tiene más popularidad es en la creación de páginas web dinámicas. El código PHP suene estar incrustado en código HTML o XHTML y precisa de un cliente que hace una petición a un servidor y de un servidor web que lo ejecuta la petición. Fue diseñada por Rasmus Lerdorf en 1995 y actualmente sigue siendo desarrollado por el grupo PHP (http://php.net/). Ejemplo de código PHP incrustado en una página XHTML editado con Notepad++:.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>ejemplo de php</title>
        <meta content="text/html; charset=utf-8" http-equiv="Content-
Type" />
    </head>
    <body>
        <?php
        $suma=0;
        for ($i=1;$i<=20;$i=$i+1)
        {   $suma=$suma+$i;
        }
        echo "<p>La suma total es: ".$suma."</p>";
        ?>
    </body>
</html>
```

Python

Es un lenguaje de programación interpretado que permite la programación orientada a objetos, con una sintaxis muy limpia que favorece que el código sea legible. Fue diseñado por Guido Van Rossum en 1991 y actualmente es administrado por la Python *Software*

Foundation (http://www.python.org/). Posee una licencia de código abierto denominada *Python Software Foundation License 1* compatible con la licencia pública general de GNU a partir de la versión 2.1.1. Ejemplo de código Python editado en Notepad++:

```
# suma los enteros del 1 al 20

def sumarEnteiros(n):
    sum=0
    for i in range (1,n+1):
            sum=sum+i

    return sum

A =sumarEnteiros(20)
print "Suma total de enteros del 1 al 20 : " + str(A)
```

JavaScript

Es un lenguaje de programación dialecto del estándar ECMAScript. Se define cómo orientado a objetos, basado en prototipos, débilmente tipado (declaración de tipos) y dinámico. Se utiliza normalmente en el lado del cliente (*client-side*) y está implementado como parte de un navegador web, aunque también existe un JavaScritpt del lado del servidor (*Server-side* JavaScript SSJS). Ejemplo de código JavaScript editado en Notepad++:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>ejemplo de JavaScript</title>
        <meta content="text/html; charset=utf-8" http-equiv="Content-
Type" />
    </head>
    <body>
        <h1>Números naturales impares hasta 9</h1>
        <script type="text/javascript">
            <!--
            var i;
            for(i=1;i<=10;i+=2)
                document.write(i+" ");
            // -->
        </script>
    </body>
</html>
```

Tarea 2. Buscar en internet los lenguajes que se repiten en un grupo de índices de popularidad dentro de los 10 primeros puestos y en el año actual y buscar las características de los lenguajes que sean diferentes a los encontrados en el año 2014.

Proceso de generación de código

La generación de código consta de los procesos de edición, compilación, y enlace. Cuando el código está finalizado, se podrá ejecutar para producir resultados.

Edición

Esta fase consiste en escribir el algoritmo de resolución en un lenguaje de programación mediante un editor de texto o una herramienta de edición incluida en un entorno de desarrollo. El código resultante se llama código fuente y el archivo correspondiente se llama archivo fuente.

Compilación

Consiste en analizar y sintetizar el código fuente mediante un compilador, para obtener, si no se encuentran errores, el código objeto o un código intermedio multiplataforma. Las personas no entienden ese código y no se puede ejecutar directamente.

Esta fase no se aplicará a los lenguajes interpretados, aunque estas pueden tener herramientas que permitan hacer un análisis léxico y sintáctico antes de pasar a ejecutarse.

Análisis

Los análisis realizados son:

- Análisis léxico en el que se comprueba que los símbolos utilizados sean correctos, incluidos los espacios en blanco.
- Análisis sintáctico en el que se comprueba que las agrupaciones de símbolos cumplan las reglas del lenguaje.
- Análisis semántico en el que se hacen el resto de comprobaciones, como, por ejemplo, que las variables utilizadas estén declaradas, o la coherencia entre el tipo de dato de una variable y el valor almacenado en ella, o la comparación del número y tipo de parámetros entre la definición y una llamada a un método.

Síntesis

La síntesis permite:

- La generación de código intermedio independiente de la máquina. Algunos lenguajes compilados como C pasan antes por una fase de preprocesamiento en la que se llevan a cabo operaciones como sustituir las constantes por el valor o incluir archivos de cabecera. Otros lenguajes como Java generan *bytecode Java* que ya podrá ser ejecutado en una JVM y otras como C# genera el código CIL que será ejecutado en el contorno CLR.
- La traducción del código intermedio anterior a código máquina para obtener el código objeto. Esta traducción lleva consigo también una optimización del código. Este nuevo código aún no está listo para ejecutarse directamente.

Enlace

Esta fase consiste en enlazar mediante un programa enlazador el archivo objeto obtenido en la compilación con módulos objetos externos para obtener, si no se encuentran errores, el archivo ejecutable.

El archivo objeto obtenido en la compilación puede tener referencias a códigos objeto externos que forman parte de bibliotecas[2] externas estáticas o dinámicas:

- Si la biblioteca es estática, el enlazador añade los códigos objeto de las bibliotecas al archivo objeto, por lo que el archivo ejecutable resultante aumenta de tamaño con relación al archivo objeto, pero no necesita nada más que el sistema operativo para ejecutarse.
- Si la biblioteca es dinámica (*Dinamic Link Library* - DLL en Windows, *Shared objects* en Linux), el enlazador añade sólo referencias a la biblioteca, por lo que el archivo ejecutable resultante apenas aumenta de tamaño con relación al archivo objeto, pero la biblioteca dinámica tiene que estar accesible cuando el archivo ejecutable se ejecute.

[2] Las bibliotecas (*library*) son colecciones de códigos objeto que tratan un tema común. Los lenguajes tienen como mínimo la biblioteca estándar. Pueden existir otras librerías como por ejemplo una para gráficos. Los programadores disponen de herramientas para crear nuevas bibliotecas.

Ejecución

La ejecución necesita de herramientas diferentes dependiendo de se el lenguaje es interpretado, compilado, de máquina virtual, o ejecución administrada.

Si el lenguaje es interpretado, será necesario el archivo fuente y el intérprete para que este vaya traduciendo cada instrucción del archivo fuente a lenguaje máquina (análisis y síntesis) y ejecutándola. Por Ejemplo: Python.

Si el lenguaje es compilado será necesario el archivo ejecutable, y en algunos casos también se necesitan bibliotecas dinámicas. Por ejemplo: C.

Si el lenguaje precisa de máquina virtual, será necesario tener el código intermedio y la máquina virtual para que esta vaya traduciendo el código intermedio el lenguaje máquina y ejecutándolo. Por ejemplo: Java o los programas para la plataforma Android. Estos últimos utilizan Java (adaptado) y la máquina virtual Dalvik (DVM) o la máquina ART (*Android Runtime*) que tienen una estructura distinta a JVM, para interpretar el código intermedio.

Tarea 3. Editar, compilar y ejecutar de forma básica código muy sencillo.