1. Linguaxes de programación e ferramentas de desenvolvemento

1.1 Introdución

Nesta parte da unidade didáctica que nos ocupa preténdense os seguintes obxectivos:

- Diferenciar linguaxes informáticas, clasificalas e identificar e caracterizar as linguaxes de programación máis populares.
- Recoñecer as características, código xerado e ferramentas utilizadas na edición, compilación, enlace e execución para linguaxes de programación compiladas, interpretadas, de máquina virtual ou de execución administrada.

1.2 Clasificación das linguaxes informáticas

Unha linguaxe informática é unha linguaxe que permite comunicarse co ordenador e está formada por un conxunto de símbolos e palabras que seguen unhas normas de sintaxe.
Non existe unha clasificación das linguaxes informáticas adoptada pola maioría dos autores senón que hai grandes diferenzas entre eles. Unha clasificación posible é: linguaxes de marcas, especificación, consulta, transformación e programación.

Linguaxes de marcas

Permiten colocar distintivos ou sinais no texto que serán interpretados por aplicacións ou procesos. Exemplos de linguaxes de marcas:

- A linguaxe **XML** (*eXtensible Markup Language*) que é unha metalinguaxe extensible pensada para a transmisión de información estruturada e que pode ser validada.
- As linguaxes **HTML** (*Hiper Text Markup Language*) ou **XHTML** (*eXtensible Hiper Text Markup Language*) =XML+HTML que serven ambas para publicar hipertexto na Word Wide Web.

Exemplo de código XML editado en NetBeans:

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

Exemplo 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">
<!-- Exemplo moi básico de XHTML con estilo externo -->
<html xmlns="http://www.w3.org/1999/xhtml">
    <head>
        <title>Exemplo básico de xhtml</title>
        <meta   content="text/html; charset=utf-8"
                http-equiv="Content-Type" />
        <link   rel="stylesheet" href="exemplo_css.css"
                type="text/css" />
    </head>
    <body>
        <h1>Cabeceira principal</h1>
        <p class="principal">Este parágrafo contén texto e unha
ligazón á
            <a href="http://academia.gal">Real Academia Galega
            </a>.
        </p>
    </body>
</html>
```

Linguaxes de especificación

Describen algo de forma precisa. Por exemplo CSS (*Cascading Style Sheets*) é unha linguaxe formal que especifica a presentación ou estilo dun documento HTML, XML ou XHTML. Exemplo de código CSS que se podería aplicar ao exemplo XHTML anterior e 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;}
```

Linguaxes de consulta

Permiten sacar ou manipular información dun grupo de información. Por exemplo, a linguaxe de consultas SQL (*Standard Query Language*) permite buscar e manipular información en bases de datos relacionais e a linguaxe XQuery permite buscar e manipular información en bases de datos XML nativas.

Exemplo de script SQL:

```
CREATE DATABASE `empresa`;

 USE `empresa`;

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

INSERT INTO `centros` VALUES
        (10,'SEDE 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_cen` 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');
```

Exemplo 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, "Toxosoutos")
return $libro
```

Linguaxes de transformación

Actúan sobre unha información inicial para obter outra nova. Por exemplo, a linguaxe XSLT (*eXtensible Stylesheet Language Tranformations*) permite describir as transformacións que se van realizar sobre un documento XML para obter outro arquivo.

Exemplo de transformación XSL sinxela editada en NetBeans, que cando actúa sobre o exemplo XML anterior obtén unha páxina HTML cunha lista dos alumnos:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Exemplo sinxelo 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="apelidos"/>,
                            <xsl:value-of select="nome"/>
                        </li>
                    </xsl:for-each>
                </ul>
            </body>
        </html>
    </xsl:template>
</xsl:stylesheet>
```

Linguaxes de programación

Permiten comunicarse cos dispositivos hardware e así poder realizar un determinado proceso e para iso poden manexar estruturas de datos almacenadas en memoria interna ou externa, e utilizar estruturas de control. Dispoñen dun léxico, e teñen que cumprir regras sintácticas e semánticas.

O **léxico** é o conxunto de símbolos que se poden usar e poden ser: identificadores (nomes de variables, tipos de datos, nomes de métodos, etcétera), constantes, variables, operadores, instrucións e comentarios.

As **regras de sintaxe** especifican a secuencia de símbolos que forman unha frase ben escrita nesa linguaxe, é dicir, para que non teña faltas de ortografía.

As **regras de semántica** definen como teñen que ser as construcións sintácticas e as expresións e tipos de datos utilizadas.

Exemplo de código Java sinxelo 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("Teclea un número enteiro entre %d e %d:",

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

Clasificación das linguaxes de programación

As linguaxes de programación poden clasificarse segundo o afastadas ou próximas que estean do hardware, por xeracións, polo paradigma da programación, pola forma de traducirse a linguaxe máquina e de executarse, e pola arquitectura cliente servidor.

Clasificación segundo a distancia ao hardware

Poden clasificarse en linguaxes de baixo e alto nivel.

Linguaxes de baixo nivel

Están baseadas directamente nos circuítos electrónicos da máquina polo que un programa escrito para unha máquina non poderá ser utilizada noutra diferente. Poden ser a linguaxe máquina ou linguaxe ensambladora.

A **linguaxe máquina** é **código binario** (ceros e uns) ou hexadecimal (números de 0 a 9 e letras de A a F) que actúa directamente sobre o hardware. É a única linguaxe que non necesita tradución xa que o procesador recoñece as instrucións directamente.

A codificación en **linguaxe ensambladora** é **mnemotécnica**, é dicir, utiliza etiquetas para describir certas operacións. É necesario traducir o código ensamblador a linguaxe máquina para que o procesador recoñeza as instrucións.

Exemplo:

Linguaxe máquina: 10110000 01100001
Linguaxe ensambladora: mov A1, #061h
Significado: Mover o valor hexadecimal 61 (97 decimal) ao rexistro do procesador chamado "A1".

Linguaxes de alto nivel

Tentan acercarse á linguaxe humana e sepáranse do coñecemento interno da máquina e por iso necesitan traducirse a linguaxe máquina. Esta tradución fai que a execución sexa máis lenta que nas linguaxes de baixo nivel pero ao non depender do procesador poden executarse en diferentes ordenadores.

Clasificación por xeracións

Poden clasificarse en 5 xeracións.

- **Primeira xeración (1GL)** formada polas linguaxes de programación utilizadas nos primeiros ordenadores: linguaxe máquina.
- **Segunda xeración (2GL)** formada polas linguaxes de baixo nivel como é a linguaxe ensambladora. Estas linguaxes son específicas para unha familia de procesadores e o hardware relacionado con ela. Aínda se sigue utilizando para programar os núcleos (*kernel*) dos sistemas operativos e os controladores dalgúns dispositivos (*device drivers*).
- **Terceira xeración (3GL)** formada pola maior parte das linguaxes de alto nivel actuais. O código é independente da máquina e a linguaxe de programación é parecida á linguaxe humana. Por exemplo, Java, C, C++, PHP, JavaScript, e Visual Basic.
- **Cuarta xeración (4GL)** formada por linguaxes e contornos deseñados para unha tarefa ou propósito moi específico como acceso a bases de datos, xeración de informes, xeración de interfaces de usuario, etcétera. Por exemplo, SQL, Informix 4GL, e Progress 4GL.
- **Quinta xeración (5GL)** formada por linguaxes nas que o programador establece o problema a resolver e as condicións a cumprir. Úsanse en intelixencia artificial, sistemas baseados no coñecemento, sistemas expertos, mecanismos de inferencia ou procesamento da linguaxe natural. Por exemplo, Prolog, Smalltalk e Lisp.

Clasificación polo paradigma da programación

Un **paradigma de programación** é unha metodoloxía ou **filosofía de programación** a seguir cun núcleo central incuestionable, é dicir, as linguaxes que utilizan o mesmo paradigma de programación utilizarán os mesmos conceptos básicos para programar. A evolución das metodoloxías de programación e a das linguaxes van parellas ao longo do tempo. Poden clasificarse en dous grandes grupos: o grupo que segue o paradigma imperativo e o que sigue o paradigma declarativo.

Paradigma imperativo

O código está formado por unha serie de pasos ou instrucións para realizar unha tarefa organizando ou cambiando valores en memoria. As instrucións execútanse de forma secuencial, é dicir, hasta que non se executa unha non se pasa a executar a seguinte. Por exemplo, Java, C, C++, PHP, JavaScript, e Visual Basic.

Dentro das linguaxes imperativas distínguese entre as que seguen a **metodoloxía estruturada** e as que seguen a **metodoloxía orientada a obxectos**.

A finais dos anos 60 naceu a metodoloxía estruturada que permitía tres tipos de estruturas no código: a **secuencial**, a **alternativa** (baseada nunha decisión) e a **repetitiva** (bucles). Os programas están formados por datos e esas estruturas.

A metodoloxía estruturada evolucionou engadindo o **módulo** como compoñente básico dando lugar á programación estruturada e modular. Os programas estaban formados por módulos ou procesos por un lado e datos por outro. Os módulos necesitaban duns datos de entrada e obtiñan outros de saída que á súa vez podían ser utilizados por outros módulos. Por exemplo, C é unha linguaxe estruturada e modular.

Nos anos 80 apareceu a **metodoloxía orientada a obxectos** que considera o obxecto como elemento básico. Esta metodoloxía popularizouse a principios dos 90 e actualmente é a máis utilizada. Por exemplo, C++ e Java son linguaxes orientadas a obxectos. Cada obxecto segue o patrón especificado nunha clase. Os programas conteñen obxectos que se relacionan ou colaboran con outros obxectos da súa mesma clase ou doutras clases. A clase está composta de atributos (datos) e métodos (procesos) e pode ter as propiedades:

- **Herdanza**. Poden declararse clases filla que herdan as propiedades e métodos da clase nai. Os métodos herdados poden sobrecargarse, é dicir, dentro da mesma clase pode aparecer definido con distinto comportamento o mesmo método con diferente firma (número de parámetros e tipo dos mesmos).
- **Polimorfismo**. Propiedade que permite que un método se comporte de diferente maneira dependendo do obxecto sobre o que se aplica.
- **Encapsulamento**. Permite ocultar detalles internos dunha clase.

Paradigma declarativo

O código indica que é o que se quere obter e non como se ten que obter, é dicir, é un conxunto de condicións, proposicións, afirmacións, restricións, ecuacións ou transformacións que describen o problema e detallan a súa solución. O programador ten que pensar na lóxica do algoritmo e non na secuencia de ordes para que se leve a cabo. Por exemplo, Lisp e Prolog son linguaxes declarativas.

Clasificación pola forma de traducirse a linguaxe máquina e executarse

Chámase **código fonte** ao código escrito nunha linguaxe de programación simbólica mediante unha ferramenta de edición. Este código gárdase nun arquivo coñecido como arquivo fonte e terá que traducirse a linguaxe máquina para poder executarse. A tradución pode facerse mediante compiladores e/ou intérpretes.

Pode considerarse que hai tres grupos de linguaxes de programación tendo en conta a forma de traducirse a linguaxe máquina e de executarse: linguaxes **compiladas**, **interpretadas**, e de **máquina virtual** ou execución administrada. Algunhas linguaxes como por exemplo Prolog poden ser consideradas como compiladas ou interpretadas xa que dispoñen de compiladores ou intérpretes.

Linguaxes compiladas

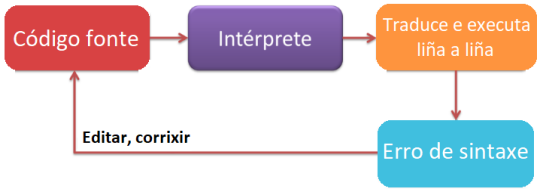
Estas linguaxes dispoñen dun compilador ou programa que traduce o código fonte a código máquina creando o arquivo executable en tempo de compilación. En tempo de execución pode lanzarse o arquivo executable na plataforma para a que serve o compilador.

Algunhas características deste tipo de linguaxes son:

- Existe un arquivo executable para a máquina real.
- O proceso de tradución faise en tempo de compilación e unha soa vez.
- A execución é moi rápida xa que o executable está en linguaxe máquina.
- O executable só funciona na plataforma para a que foi creado.
- O usuario que executa non ten que coñecer o código fonte, só ten o código executable que non é manipulable facilmente para obter o código fonte, e como consecuencia o programador ten o código fonte máis protexido.
- O executable non se xerará se existen erros léxicos, sintácticos ou semánticos.
- Interromper a execución pode ser difícil para o sistema operativo e pode afectar á plataforma.
- A modificación do código fonte implica volver xerar o arquivo executable.

Linguaxes interpretadas

Estas linguaxes precisan dun interprete ou programa en memoria para que en tempo de execución se vaia traducindo o código fonte a linguaxe máquina.

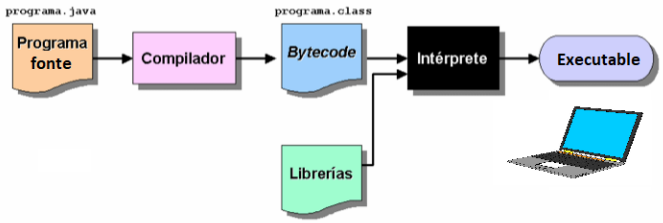


Algunhas características deste tipo de linguaxes son:

- Non existe un arquivo executable.
- O proceso de tradución faise cada vez que se executa.
- A execución é lenta xa que ao proceso de execución ten que sumarse o de tradución.
- arquivo pode executarse en diferentes plataformas sempre que exista intérprete para elas.
- O usuario utiliza o código fonte para executar o programa.
- Poden aparecer na execución erros de tipo léxico, sintácticos ou semánticos.
- Interromper a execución só afecta normalmente ao intérprete e non á plataforma.
- Pode executarse en calquera plataforma sempre que haxa intérprete para ela.
- A modificación do código fonte non require ningunha operación extra antes de executar o programa.

Linguaxes de máquina virtual ou de execución administrada

Estas linguaxes precisan inicialmente dun compilador que en tempo de programación traduce o código fonte a un código intermedio multiplataforma, e a continuación necesita doutro software que traduza ese código intermedio a código máquina.



No caso de linguaxes de máquina virtual como Java ocorre que:

- En tempo de compilación o compilador Java xera o código intermedio chamado bytecode Java independente da plataforma.
- En tempo de execución, necesita dunha máquina virtual Java (JVM) que interprete o bytecode. Esta máquina virtual é un software que simula unha máquina real co seu propio sistema operativo e que fai de intermediaria coa máquina real, polo que as máquinas virtuais son diferentes para Linux, Windows, Mac e Solaris.

No caso de linguaxes de execución administrada como as da plataforma .NET[1] de Microsoft ocorre que:

- En tempo de compilación, o compilador xera o código CIL (*Common Intermediate* Language) independente da plataforma.
- Para executar é necesario ter no equipo cliente a versión de .NET Framework (e por tanto de CRL- Common Runtime Language) adecuada e entón faise a última fase da compilación, na que o CRL traducirá o código intermedio a código máquina mediante un compilador JIT (*Just In Time*).

Algunhas características deste tipo das linguaxes de máquina virtual ou de execución administrada son:

- Existe un código intermedio que se executa nun software específico pero non é un executable.
- O proceso de tradución inicial faise antes da execución e o de tradución final faise cada vez que se executa.
- A execución non é tan rápida como nas linguaxes compiladas pero é máis rápida que nas linguaxes interpretadas.
- O arquivo pode executarse en diferentes plataformas sempre que exista o software específico para iso.
- O usuario non ten o código fonte, só o código intermedio que non é manipulable facilmente para obter o código fonte polo que o programador ten o código fonte máis protexido.
- Os erros de tipo léxico, sintáctico ou semántico detéctanse na fase de compilación, e se existen non se xerará o código intermedio.
- Interromper a execución só afecta normalmente ao intérprete e non á plataforma.
- A modificación do código fonte implica volver repetir o proceso de compilación.

Clasificación na arquitectura cliente-servidor

A arquitectura cliente-servidor consiste basicamente nun programa cliente que realiza peticións a un servidor. Esta arquitectura é máis útil cando o cliente e o servidor están comunicados mediante unha rede, aínda que tamén se pode aplicar cando están na mesma máquina. Nesta arquitectura diferénciase entre linguaxes que se executan:

- do lado do servidor como PHP.

[1] Esta plataforma é un conxunto de tecnoloxías de software que ten un contorno de traballo (.NET Framework) e varias linguaxes de programación como por exemplo VisualBasic.NET, C# ou C++. O contorno de traballo, incluído no sistema operativo Windows, inclúe un contorno de execución administrado chamado CRL (*Common Runtime Language*) que executa código e proporciona servizos que facilitan o proceso de desenvolvemento.

- do lado do cliente como JavaScript.

Para explicar isto pódese supoñer un navegador cliente e un servidor web con intérprete de PHP e a situación mostrada nos seguintes pasos:

- Un usuario dende un navegador cliente pide a un servidor Web unha páxina HTML que ten incrustado código PHP e código JavaScript.
- O servidor Web procesa a petición, interpreta o código PHP, execútao colocando o resultado da execución no sitio onde estaba o código PHP e devolve ao cliente unha páxina web con código HTML e JavaScript. O servidor Web debería de executar as ordes relativas ao manexo dunha base de datos nun servidor de base de datos se o código PHP tivera ese tipo de instrucións.
- O navegador cliente interpreta o codigo JavaScript e mostra a páxina ao usuario.

O usuario pode interactuar coa páxina ocorrendo que cada vez que fai peticións ao servidor recárgase a páxina completamente coa resposta do servidor.

Existe unha técnica denominada AJAX (*Asynchronous JavaScript And XML*), que permite que JavaScript procese algún evento iniciado polo usuario facendo unha petición ao servidor que este responde co resultado en XML. JavaScript procesa o resultado XML actualizando seccións da páxina sen ter que recargala totalmente e logrando así unha interacción asíncrona entre servidor e cliente.

Linguaxes máis difundidas

Para saber cales son as linguaxes máis difundidas poden consultarse os seguintes índices ou clasificacións:

- Índice Tiobe (https://www.tiobe.com/tiobe-index//) que basea a clasificación das linguaxes de programación no número de enxeñeiros cualificados en cada linguaxe en todo o mundo, cursos de linguaxes ofertados, provedores que traballan sobre as linguaxes, buscas realizadas en Google, Bing, Yahoo, Wikipedia, Amazon, YouTube e Baidu e unha serie de premisas como por exemplo que a linguaxe exista como linguaxe de programación en Wikipedia e que teña polo menos 10000 visitas en Google.
- Índice PYPL ou *PopularitY of Programming Language index* (https://pypl.github.io/PYPL.html) e basea a clasificación das linguaxes de programación no análise da frecuencia de busca de titoriais ou guías das linguaxes de programación en Google utilizando Google Trends.
- A clasificación Redmonk (https://redmonk.com/sogrady/2021/08/05/language-rankings-6-21/) que non basea a clasificación nos buscadores senón nos proxectos albergados no repositorio GitHub e nas preguntas da web de StackOverflow

orientada a programadores. Inclúe linguaxes informáticas e non só linguaxes de programación.

- A clasificación Trendyskills (http://trendyskills.com/) baséase nas ofertas de emprego para as linguaxes de programación en España, USA, UK, Alemaña, Suecia, Países Baixos, Irlanda, Suíza, Austria, Bélxica, Finlandia, República Checa e Grecia e inclúe linguaxes informáticas e non só linguaxes de programación.

As linguaxes comúns a todos estes índices no ano 2021 nos 10 primeiros postos son: C++, Java, C#, PHP, Python e JavaScript.

10 Linguaxes de programación máis populares en 2021			
Índice Tiobe setembro 2021	**Índice PYPL setembro 2021**	**Clasificación Redmonk xuño 2021**	**Clasificación Trendyskills setembro de 2021**
C	Python	JavaScript	Java
Python	Java	Python	JavaScript
Java	JavaScript	Java	Python
C++	C#	PHP	C++
C#	PHP	CSS	HTML
Visual Basic	C/C++	C++	C#
JavaScript	R	C#	TypeScript
Assembly language	TypeScript	TypeScript	PHP
PHP	Swift	Ruby	Go
SQL	Objective-C	C	XML

C

C é unha linguaxe creada por Dennis Ritchie en 1972 nos laboratorios Bell para codificar o sistema operativo UNIX. Está catalogada como unha linguaxe de alto nivel, estruturada e modular, pero con moitas características de baixo nivel como por exemplo utilizar punteiros para facer referencia a unha posición física da memoria RAM. Estas características posibilitan traballar moi cerca da máquina pero os programas son máis complicados e propensos a ter máis erros. Influíu no deseño das linguaxes C++, C#, Java, PHP e JavaScript entre outras.

Exemplo de código C:

```c
/*
 * Programa C que suma os números enteiros do 1 ó 20
 */
#include <stdio.h>
#include <stdlib.h>
int main(int argc, char** argv) {
    int i;              /* contador de números enteiros */
    long int suma;      /* sumador dos números enteiros */
    int final;
    suma = 0;
    for (i = 1; i <=20; i++) {
        suma = suma + i;
    }
    printf("Programa C\nA suma total e:  %ld\n", suma);
    printf("\nTeclee calquera numero para finalizar...");
    scanf("%d",&final);
    fflush(stdin);
    return (EXIT_SUCCESS);
}
```

C++

Deseñada en 1980 por Bjarne Stroustrup para estender C con mecanismos que permitan a POO (Programación Orientada a Obxectos). Exemplo de código C++ editado en NetBeans:

```cpp
/*
 * Programa C++ que suma os números enteiros do 1 ó 20
 */
#include <iostream>
int main(int argc, char**argv) {
    int i;              /* contador de números enteiros */
    long int suma;      /* sumador dos números enteiros */
    suma = 0;
    for (i = 1; i <=20; i++) {
        suma = suma + i;
    }
    std::cout <<"A suma total e "<< suma<< std::endl;
    return 0;
}
```

Java

É unha linguaxe de programación orientada a obxectos desenvolvida por Sun Microsystems en 1995. Toma moita da súa sintaxe de C e C++ pero cun modelo de obxectos máis simple e elimina as ferramentas de baixo nivel que se utilizaban en C.

Exemplo de código Java editado en NetBeans:

```
package sumaenteiros;
/*
 * File: Main.java
 * Author: profesor
 * Obxectivo: visualizar a suma dos 20 primeiros números naturais
 */
public class Main {

    public static void main(String[] args) {
        int suma=0;     /* sumador dos números enteiros */
        for (int i=1;i<=20;i++)
        {
            suma=suma+i;
        }
        System.out.printf("Exemplo Java\n");
        System.out.printf("Suma dos 20 primeiros números naturais =
%d\n",suma);
    }
}
```

C#

É unha linguaxe de programación orientada a obxectos desenvolvida por Microsoft no ano 2000 como parte da plataforma .NET. Visual C# proporciona un editor de código avanzado, deseñadores de interface de usuario, e numerosas ferramentas para facilitar o desenvolvemento de aplicacións en C# e .NET Framework.

Exemplo de código C# creado con Visual Studio e editado con Notepad++:

```
using System;

namespace exemplo_csharp_consola
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Ola Mundo");
        }
    }
}
```

PHP

PHP *(Hypertext PreProcessor)* é unha linguaxe interpretada do lado do servidor que se pode usar para crear calquera tipo de programa pero que onde ten máis popularidade é na creación de páxinas web dinámicas. O código PHP soe estar incrustado en código HTML ou XHTML e precisa dun cliente que fai unha petición a un servidor e dun servidor web que executa a petición. Foi deseñada por Rasmus Lerdorf en 1995 e actualmente segue sendo desenvolvida polo grupo PHP (http://php.net/).

Exemplo de código PHP incrustado nunha páxina 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>exemplo 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+$i;
        }
        echo "<p>A suma total é: ".$suma."</p>";
        ?>
    </body>
</html>
```

Python

É unha linguaxe de programación interpretada que permite a programación orientada a obxectos, cunha sintaxe moi limpa que favorece que o código sexa lexible. Foi deseñada por Guido Van Rossum en 1991 e actualmente é administrada pola *Python Software Foundation* (http://www.python.org/). Posúe unha licenza de código aberto denominada *Python Software Foundation License 1* (PSFL) compatible coa licenza pública xeral de GNU a partir da versión 2.1.1.

Exemplo de código Python editado en Notepad++:

```
# suma os enteiros do 1 ao 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 enteiros do 1 ao 20 : " + str(A)
```

JavaScript

É unha linguaxe de programación dialecto do estándar ECMAScript. Defínese como orientada a obxectos, baseada en prototipos, debilmente tipada (declaración de tipos) e dinámica. Utilízase normalmente no lado do cliente (*client-side*) e está implementada como parte dun navegador web aínda que tamén existe un JavaScript do lado do servidor (*Server-side* JavaScript ou SSJS).

Exemplo 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>exemplo de JavaScript</title>
        <meta content="text/html; charset=utf-8" http-equiv="Content-
Type" />
    </head>
    <body>
        <h1>Números naturais impares ata 9</h1>
        <script type="text/javascript">
            <!--
            var i;
            for(i=1;i<=10;i+=2)
                document.write(i+" ");
            // -->
        </script>
    </body>
</html>
```

Proceso de xeración de código

A xeración de código consta dos procesos de edición, compilación, e enlace. Cando o código estea finalizado, poderase executar para producir resultados.

Edición

Esta fase consiste en escribir o algoritmo de resolución nunha linguaxe de programación mediante un editor de texto ou unha ferramenta de edición incluída nun contorno de desenvolvemento. O código resultante chámase código fonte e o arquivo correspondente chámase arquivo fonte.

Compilación

Consiste en analizar e sintetizar o código fonte mediante un compilador, para obter, se non se atopan erros, o código obxecto ou un código intermedio multiplataforma. As persoas non entenden ese código e non se pode executar directamente.

Esta fase non se aplicará ás linguaxes interpretadas aínda que estas poden ter ferramentas que permitan facer unha análise léxica e sintáctica antes de pasar a executarse.

Análise

As análises realizadas son:

- **Análise léxica** na que se comproba que os símbolos utilizados sexan correctos, incluídos os espazos en branco.
- **Análise sintáctica** na que se comproba que as agrupacións de símbolos cumpran as regras da linguaxe.
- **Análise semántica** na que se fan o resto de comprobacións, como por exemplo, que as variables utilizadas estean declaradas, ou a coherencia entre o tipo de dato

dunha variable e o valor almacenado nela, ou a comparación do número e tipo de parámetros entre a definición e unha chamada a un método.

Síntese

A síntese permite:

- A xeración de código intermedio independente da máquina. Algunhas linguaxes compiladas como C pasan antes por unha fase de preprocesamento na que se levan a cabo operacións como substituír as constantes polo valor ou incluír arquivos de cabeceira. Outras linguaxes como Java xeran *bytecode Java* que xa poderá ser executado nunha JVM e outras como C# xeran o código CIL que será executado no contorno CLR.
- A tradución do código intermedio anterior a código máquina para obter o código obxecto. Esta tradución leva consigo tamén unha optimización do código. Este novo código aínda non está listo para executarse directamente.

Enlace

Esta fase consiste en enlazar mediante un programa enlazador o arquivo obxecto obtido na compilación con módulos obxecto externos para obter, se non se atopan erros, o arquivo executable.

O arquivo obxecto obtido na compilación pode ter referencias a códigos obxecto externos que forman parte de bibliotecas[2] externas estáticas ou dinámicas:

- Se a biblioteca é estática, o enlazador engade os códigos obxecto das bibliotecas ao arquivo obxecto, polo que o arquivo executable resultante aumenta de tamaño con relación ao arquivo obxecto, pero non necesita nada máis que o sistema operativo para executarse.
- Se a biblioteca é dinámica (*Dinamic Link Library* - DLL en Windows, *Shared objects* en Linux), o enlazador engade só referencias á biblioteca, polo que o arquivo executable resultante apenas aumenta de tamaño con relación ao arquivo obxecto, pero a biblioteca dinámica ten que estar accesible cando o arquivo executable se execute.

Execución

A execución necesita de ferramentas diferentes dependendo de se a linguaxe é interpretada, compilada ou de máquina virtual ou execución administrada.

[2] As bibliotecas (*library*) son coleccións de códigos obxecto que tratan un tema común. As linguaxes teñen como mínimo a biblioteca estándar. Poden existir outras bibliotecas como por exemplo unha para gráficos. Os programadores dispoñen de ferramentas para crear novas bibliotecas.

Se a linguaxe é interpretada, será necesario o arquivo fonte e o intérprete para que este vaia traducindo cada instrución do arquivo fonte a linguaxe máquina (análise e síntese) e executándoa. Por Exemplo: PHP.

Se a linguaxe é compilada será necesario o arquivo executable, e nalgúns casos tamén se necesitan bibliotecas dinámicas. Por exemplo: C.

Se a linguaxe precisa de máquina virtual, será necesario ter o código intermedio e a máquina virtual para que esta vaia traducindo o código intermedio a linguaxe máquina e executándoo. Por exemplo: Java ou os programas para a plataforma Android. Estes últimos utilizan Java (adaptado) e a máquina virtual Dalvik (DVM) ou a máquina ART (*Android Runtime*) que teñen unha estrutura distinta a JVM, para interpretar o código intermedio.