Sistemas y Aplicaciones Informáticas

Tema 26. Lenguajes de Programación. Tipos y Características.

1. Ámbito de docencia.

- Sistemas informáticos monousuario y multiusuario (ASI 1).
- Sistemas informáticos multiusuario y en red (DAI 1).
- Sistemas operativos en entornos monousuario y multiusuario (ESI 1).

2. Lenguajes de programación.

2.1. Introducción.

2.1.1. Programas. Lenguajes de programación.

- Un programa es un conjunto de instrucciones ejecutables por un ordenador, y está formado por algoritmos y estructuras de datos que producen un determinado resultado.
- Los programas están expresados en un determinado lenguaje de programación, que consiste en un conjunto de símbolos y reglas que permiten comunicar esta información al ordenador.

2.1.2. Criterios de evaluación de los lenguajes de programación.

- **Facilidad de lectura y comprensión**. Los programas deben ser corregidos y modificados, por lo tanto, el lenguaje debe facilitar la comprensión de los programas una vez escritos.
- **Facilidad de codificación**. Debe ser simple utilizar el lenguaje para desarrollar programas que se ajusten al tipo de problemas para el que está orientado.
- **Fiabilidad**. Un lenguaje es fiable si funciona bien en cualquier condición. Depende de la comprobación de tipos de dato, del control de excepciones y del manejo de la memoria.
- **Entorno de programación**. Se refiere a la existencia de herramientas para el desarrollo.
- **Portabilidad de programas**. Un mismo programa debe funcionar en ordenadores diferentes.
- **Coste**. Relacionado con el aprendizaje, codificación, ejecución, fiabilidad y mantenimiento.

2.1.3. Elementos léxicos, sintácticos y semánticos.

- **Léxicos**. Son el conjunto de símbolos permitidos en el lenguaje. Pueden dar lugar a:
  - *Identificadores*. Nombres simbólicos que se da a ciertos elementos de programación, por ejemplo nombres de variables, tipos, módulos, etc.
  - *Operadores*. Símbolos que representan operaciones entre variables y constantes.
  - *Palabras reservadas*. Símbolos especiales que representan instrucciones de procesamiento y definiciones de elementos de programación.
  - *Comentarios*. Símbolo que se utiliza para documentar los programas.
  - *Delimitadores*. Símbolos utilizados para señalar el principio o el final de una unidad sintáctica, como un enunciado o una expresión.
- **Sintácticos**. Son el conjunto de reglas que indican la forma de realizar las construcciones del lenguaje. Las reglas sintácticas pueden tener elementos terminales (símbolos léxicos) y elementos no terminales (construcciones gramaticales intermedias). Pueden expresarse mediante:
  - *Notación BNF*. Tiene la forma <elemento no terminal>::= definición1 | definición2 | ...
  - *Diagrama sintáctico*. Representación gráfica de la sintaxis. Los elementos no terminales se inscriben en un rectángulo y los elementos terminales en elipses.
- **Semánticos**. Son el conjunto de reglas que permiten determinar el significado de cualquier construcción del lenguaje, y de las expresiones y tipos de datos utilizadas.

2.2. Técnicas de programación.

2.2.1. Programación lineal.

- Es un tipo de programación que no sigue ningún método, rompiendo la secuencia del programa incondicionalmente (GOTO) y estableciendo varios puntos de entrada y salida del código.
- Esta forma de programar complica la legibilidad del código, y por tanto incrementa la dificultad en el mantenimiento de los programas y en la documentación de las aplicaciones.

2.2.2. Programación estructurada.

- Se basa en el teorema de Böhm y Jacopini, por el cual todo programa con un único punto de entrada y de salida de código puede escribirse utilizando sólo las tres estructuras siguientes:
  - *Secuenciales*. Formadas por la sucesión ordenada de varias instrucciones.
  - *Alternativas*. Basadas en la evaluación de una expresión lógica o relacional para controlar la realización de ciertas instrucciones y alterar de este modo el orden secuencial del programa.
  - *Iterativas*. Basadas en la evaluación de una expresión lógica o relacional para repetir la realización de ciertas instrucciones y alterar de este modo el orden secuencial del programa.
- Además se caracteriza por aplicar la técnica de descomposición de tareas en otras más pequeñas denominada diseño descendente, que consiste en una serie de descomposiciones sucesivas del problema inicial, estableciendo una serie de niveles de mayor a menor complejidad que den solución al problema. Los niveles o procedimientos se estructuran jerárquicamente de manera que un nivel y su inmediato inferior se relacionan mediante entradas y salidas de información.

2.2.3. Programación modular.

- Es una técnica complementaria a la programación estructurada, consistente en aplicar el diseño descendente y separar cada tarea sencilla en un subprograma independiente denominado módulo, que reciba una serie de datos y devuelva un resultado o realice alguna acción.
- Los módulos pueden ser invocados desde cualquier parte del código y su funcionamiento no depende del resto del programa, por lo que es más fácil encontrar los errores y realizar el mantenimiento. El programador se concentra en codificar cada módulo haciendo más sencilla esta tarea. Para dar lugar a la aplicación final debe realizarse una integración de módulos.

3. Tipos y características.

3.1. Clasificación de los lenguajes de programación.

3.1.1. Según su etapa de generación.

3.1.1.1. Lenguajes máquina (primera generación).

- Consiste en un conjunto limitado de instrucciones y una serie de datos codificados en binario que el procesador es capaz de ejecutar directamente.
- Un programa escrito en cualquier lenguaje de programación debe transformarse en lenguaje máquina para poder ser ejecutado, lo que hace necesaria la utilización de traductores.
- Se caracteriza por lo siguiente:
  - Las instrucciones están definidas en los circuitos del procesador y realizan operaciones simples. Los datos y el destino de las instrucciones de salto de ejecución se referencian por medio de su posición en la memoria, y por tanto es necesario conocer su dirección.

- El lenguaje máquina es particular de cada procesador y por tanto no es portable. Es muy difícil y costoso de programar y mantener, pero el código es totalmente eficiente.

3.1.1.2. Lenguajes ensambladores (segunda generación).

- Son lenguajes simbólicos que asignan a cada instrucción del lenguaje máquina un código nemotécnico formado por caracteres alfanuméricos.
- Se caracterizan por lo siguiente:
  - Se facilita la escritura y depuración de los programas pero no los acorta. Se emplean etiquetas para referenciar cualquier parte del código y nombres para referenciar a los datos, no siendo por tanto necesario conocer su dirección. También permite el uso de comentarios.
  - Al tratarse de transformaciones del lenguaje máquina, los lenguajes ensambladores son también particulares de cada procesador y por tanto no son portables.
  - Es menos costoso que el lenguaje máquina, pero igualmente exige un total conocimiento del hardware. Está indicado sobre todo para programar tareas relacionadas con el control de los periféricos y la gestión de la memoria.
- Debido a que la programación de cualquier tarea debe ser descompuesta en multitud de pasos elementales, para aumentar la eficiencia en la programación se recurre a crear instrucciones denominadas macroinstrucciones que agrupen a varias instrucciones de código máquina. El lenguaje así construido se denomina macroensamblador.

3.1.1.3. Procedimentales (tercera generación).

- Son lenguajes más cercanos al lenguaje humano en los cuales la solución al problema se describe paso a paso mediante un conjunto de instrucciones más flexibles y potentes.
- Se caracterizan por lo siguiente:
  - Son independientes del hardware, y por tanto son portables. En la práctica, esta característica está limitada por la gran diversidad de versiones que existen de cada lenguaje.
  - Una instrucción en estos lenguajes da lugar a varias instrucciones en lenguaje máquina. Son más lentos y menos eficientes que los lenguajes ensambladores ya que deben ser traducidos a código máquina, pero son más sencillos de codificar y mantener.
  - A diferencia de los lenguajes ensambladores, soportan la programación estructurada.
- Entre los lenguajes procedimentales se pueden destacar actualmente Modula-2, C, C++ y Java.

3.1.1.4. No procedimentales (cuarta generación).

- Son lenguajes en los que el programador no especifica el procedimiento a seguir, puesto que el propio lenguaje es capaz de indicar al ordenador cómo debe ejecutar el programa.
- Se caracterizan por lo siguiente:
  - El programador debe definir una serie de parámetros que, mediante la utilización de un conjunto de herramientas del lenguaje, permite generar un programa de aplicación.
  - Son más fáciles de utilizar que los de tercera generación, ya que suelen incluir interfaces gráficos y capacidades de gestión avanzadas, pero consumen muchos más recursos.
  - No requieren una capacitación especial por parte del programador, mejorando su productividad, y están diseñados para resolver problemas específicos.
- En los lenguajes de cuarta generación están incluidos:

    - Lenguajes de presentación, como lenguajes de consultas y generadores de informes.
    - Lenguajes especializados, como hojas de cálculo y lenguajes de bases de datos.
    - Generadores de aplicaciones que definen, actualizan y obtienen datos de una base de datos.
    - Lenguajes utilizados para generar el código de una aplicación.
- Entre este tipo de lenguajes se pueden destacar actualmente SQL, Visual Basic y Forms.

3.1.1.5. Naturales (quinta generación).

- Son lenguajes cuyos programas están constituidos por hechos, reglas, restricciones, ecuaciones, transformaciones, u otras propiedades que debe cumplir la solución al problema.
- Se caracterizan por lo siguiente:
    - A partir de la información introducida por el programador, el sistema debe deducir un esquema que incluya una secuencia de evaluaciones para calcular la solución.
    - Están orientados a aplicaciones en inteligencia artificial, es decir, sistemas basados en el conocimiento, sistemas expertos o procesamiento del lenguaje natural.
- Estos lenguajes se encuentran aún en fase de desarrollo, y están basados en LISP y PROLOG.

3.1.2. Según su cercanía a la máquina.

- **Bajo nivel**. Son aquellos que se encuentran más cercanos a la máquina, y por tanto más alejados del lenguaje humano. Están formados por los lenguajes máquina y los lenguajes ensambladores.
- **Alto nivel**. Son todos los demás, y se caracterizan principalmente por su independencia de la máquina, por su necesidad de traducción al lenguaje máquina y su cercanía al lenguaje humano.

3.1.3. Según su paradigma de programación.

3.1.3.1. Imperativos.

- Se basan en un modelo de ordenador como una máquina que almacena datos y un conjunto de instrucciones ejecutadas secuencialmente que se encargan de modificar dichos datos. Están formados por sentencias agrupadas en procedimientos, cada uno de los cuales realiza una tarea concreta y se relaciona con el resto del programa mediante llamadas con paso de parámetros.
- En su forma más pura sólo soportan la modificación de datos mediante la utilización de variables y sentencias de asignación, sentencias de salto condicional y de salto incondicional. Son más eficientes porque se asemejan al modo de funcionamiento de la máquina.
- Entre los lenguajes imperativos se pueden destacar actualmente Modula-2 y C.

3.1.3.2. Orientados a objetos.

- Es un tipo de programación de un nivel de abstracción mayor, que consiste en modelar la realidad como un conjunto de objetos que interactúan entre sí para resolver un problema.
- Las características básicas comunes que presentan los lenguajes orientados a objetos son:
    - *Encapsulamiento*. Las propiedades y el comportamiento de los objetos están definidos en las clases, formadas por estructuras de datos y algoritmos denominadas atributos y métodos. Un objeto es una instancia de una determinada clase y tiene su propio conjunto de datos. Los métodos se aplican a un objeto concreto y son propias de la clase a la que pertenece.
    - *Ocultación*. Para aumentar la modularidad y como medida de protección, un objeto se considera como una caja negra en la que se puede introducir o de la que se puede extraer información sin necesidad de conocer su funcionamiento interno.

- *Envío de mensajes*. Consiste en nombrar y activar con los parámetros adecuados uno de los métodos del objeto al que se envía el mensaje. Es la manera que tienen los objetos de relacionarse, y es equivalente a las llamadas a procedimientos de los lenguajes imperativos.
- *Herencia*. Es un mecanismo mediante el cual pueden crearse clases a partir de otras, transmitiéndose todos los atributos y todos los métodos de clases padres a clases hijas, con la ventaja de que en estas últimas pueden añadirse más atributos y métodos.
- *Polimorfismo*. Es la capacidad que tiene un objeto de una determinada clase de hacerse pasar por un objeto de una clase ancestro. Esta característica, junto a la posibilidad de las clases hijas de redefinir los métodos heredados, permiten la reutilización de código.

– Entre los lenguajes orientados a objetos se pueden destacar actualmente C++ y Java.

3.1.3.3. Funcionales.

– Se trata de lenguajes de programación sin asignaciones, basados en el concepto matemático de función. Consisten en combinar funciones para conseguir funciones más complejas hasta llegar a la función que es el programa. Los valores de las funciones son también funciones.

– Ejemplos de estos lenguajes son LISP y SCHEME.

3.1.3.4. Lógicos.

– Son lenguajes en los que se especifican un conjunto de hechos y una serie de reglas que permiten la deducción de otros hechos. El sistema utiliza esa información para encontrar la solución.

– La programación lógica desde la perspectiva del programador consiste en establecer correctamente todos los hechos y reglas, ya que el cálculo está implícito.

– El lenguaje lógico más representativo es PROLOG, basado en la lógica de predicados.

3.1.3.5. Concurrentes.

– Son lenguajes cuyo elemento fundamental es el proceso, que es una instancia de un programa que está siendo ejecutado por un ordenador junto con sus datos asociados y los recursos que utiliza. Un hilo de un programa concurrente es cada uno de los flujos secuenciales de control independientes especificados en dicho programa.

– Los programas concurrentes dan lugar a un proceso con varios hilos de ejecución. La concurrencia puede ser física, cuando existe más de un procesador y varios hilos de un mismo programa se ejecutan realmente de forma simultánea, o lógica, cuando existe un único procesador y hay que repartir su tiempo entre los distintos hilos. Hay dos modelos de ejecución:

- *Transformativo*. Cuando el fin consiste en transformar los datos de entrada en valores de salida apropiados, y la concurrencia se aplica para acelerar el proceso.
- *Reactivo*. Cuando se producen sucesos externos asíncronos, como en los sistemas de tiempo real, y la concurrencia se aplica para que el programa reaccione ante dichos sucesos.

– Un lenguaje de programación es concurrente si posee las estructuras necesarias para definir y manejar diferentes hilos de ejecución dentro de un programa, por ejemplo Java y ADA.

3.1.4. Según su traducción a lenguaje máquina.

3.1.4.1. Compilados.

– Son aquellos cuyos programas fuente son traducidos en su totalidad a lenguaje máquina antes de ser ejecutados. Esta operación se realiza en dos pasos:

- *Compilación*. Se realiza un análisis léxico, sintáctico y semántico del programa fuente, seguido de un proceso de optimización de código. Esto da lugar a código máquina no ejecutable denominado código objeto, ya que en compilación no es posible determinar las direcciones de memoria de las variables y las instrucciones, debido a lo siguiente:
  - Un programa fuente puede estar dividido en varios ficheros, que se interrelacionan de tal forma que pueden compartir algunas variables entre ellos, así como pueden utilizarse en unos ficheros funciones cuyo código se encuentra escrito en otros.
  - Todos los lenguajes disponen de librerías que contienen funciones para utilizar. Las librerías se encuentran en código objeto y no pueden tener asignadas las direcciones de las variables ni de las instrucciones ya que pueden formar parte de cualquier programa.
- *Enlazado*. Después de compilar todos los programas fuente, es necesario combinar los módulos objeto correspondientes en un solo programa:
  - A las variables se les asigna su dirección en memoria y todas las referencias a dichas variables tienen que apuntar a sus correspondientes direcciones de memoria. Las funciones y los procedimientos obtienen una posición de memoria también.
  - Cuando el sistema soporta varios programas al mismo tiempo en la memoria, las direcciones de los símbolos no pueden ser fijas pues se producirían conflictos entre unos programas y otros. En este caso se asignan a los símbolos direcciones relativas respecto de la dirección que el sistema operativo le da al programa en el momento de ser cargado en memoria. Se dice entonces que el programa es reubicable.

– El proceso de compilación puede ser lento, puesto que todo el código debe ser correcto antes de poder ejecutarlo. Sin embargo el lenguaje máquina generado es eficiente porque está optimizado.
– Entre los lenguajes compilados se pueden destacar actualmente C y C++.

3.1.4.2. Interpretados.

– Son aquellos en los que cada línea del programa fuente se traduce a lenguaje máquina y se ejecuta inmediatamente, sin crearse por tanto un programa objeto para posteriores ejecuciones.
– Son más lentos y menos eficientes que los compilados porque el código no se analiza de forma global, y necesitan de una herramienta traductora (máquina virtual) cada vez que se ejecutan. Sin embargo permiten una rápida depuración de los programas, ya que puede ejecutarse parcialmente el código fuente aunque no esté finalizado.
– Entre los lenguajes interpretados se pueden destacar actualmente Visual Basic y los lenguajes de script en entorno Web como JavaScript, PHP y ASP.

3.1.4.3. Híbridos.

– Son aquellos cuyos programas fuente son traducidos en su totalidad a un lenguaje intermedio, mas cercano al lenguaje máquina que al código fuente, que posteriormente es interpretado.
– Combinan las ventajas de los métodos de traducción anteriores, y además garantiza la portabilidad puesto que el código intermedio es igual en todas las plataformas, sólo hay que utilizar el intérprete adecuado para cada una de ellas.
– El ejemplo más representativo es Java. Al compilarlo genera un código denominado bytecode que posteriormente es ejecutado mediante la máquina virtual y el entorno de ejecución de Java.

3.1.5. Según su ámbito de aplicación.

- **Propósito general**. Son lenguajes que pueden utilizarse para solucionar una amplia variedad de problemas. Entre ellos se pueden destacar C, C++ y Java.
- **Cálculo científico**. Se trata de lenguajes que facilitan la programación de algoritmos, es decir están orientados al cálculo matemático. Los programas desarrollados son rápidos pudiendo realizar muchas operaciones por segundo y ocupan poco espacio en memoria. A cambio suelen dar pocas facilidades al programador para crear una interfase amigable con el usuario, dibujar gráficos o manejar ficheros de datos. Se pueden destacar FORTRAN y Matlab.
- **Gestión**. Son lenguajes especializados en manejar gran cantidad de información que se encuentra almacenada en el disco. Permiten desarrollar de forma relativamente sencilla y en poco tiempo aplicaciones de gestión de empresas tales como programas de contabilidad, nóminas, facturación, gestión de almacenes, etc. Se pueden destacar COBOL y RPG.
- **Inteligencia artificial**. Facilitan la programación de aplicaciones tales como sistemas expertos. Son útiles para programar diversas técnicas de representación del conocimiento utilizadas en inteligencia artificial. Se pueden destacar PROLOG y LISP.
- **Programación web**. Son lenguajes interpretados como JavaScript o VBScript, o lenguajes especiales para uso en servidores como ASP, JSP o PHP. Estos lenguajes se mezclan con el código HTML para crear páginas web en el cliente y manejar información en el servidor.

3.1.6. Según su administración de memoria.

- **Estáticos**. Aquellos en los que los requisitos de memoria de los programas se calculan durante la compilación y no admiten recursividad ni estructuras dinámicas. Por ejemplo FORTRAN.
- **Basados en pila**. Aquellos en los que en tiempo de compilación se asigna un espacio de memoria que contiene la pila donde se almacenan las variables locales y el heap donde se almacenan las estructuras dinámicas. Ambas pueden variar dinámicamente de tamaño durante la ejecución, por tanto permiten la recursividad y las estructuras dinámicas. Por ejemplo C.
- **Dinámicos**. Aquellos en los que no se puede saber a priori la cantidad de memoria que utilizará el programa. El programa puede crear y destruir estructuras de datos en cualquier lugar del programa. Por ejemplo PROLOG y LISP.

3.2. Historia y evolución.

- **1945**. Presentación de los principios de la máquina de propósito general de Von Neumann. Los primeros lenguajes de programación son los lenguajes máquina de cada procesador.
- **1951**. Los ensambladores surgen como traductores de representaciones simbólicas nemotécnicas del lenguaje máquina (lenguajes ensambladores) al propio lenguaje máquina.
- **1956**. Aparece FORTRAN como el primer lenguaje de alto nivel compilado. Es un lenguaje imperativo orientado a cálculo científico. La última versión es el FORTRAN 90.
- **1957**. Surge LISP como el primer lenguaje funcional. Se utiliza en Inteligencia Artificial, y su descendiente actual es COMMON LISP.
- **1959**. Aparece COBOL como el primer lenguaje estandarizado. Muy utilizado en aplicaciones de gestión, contabilidad y bases de datos, pero con muy poca influencia posterior.

- **1960**. Surge ALGOL a partir de un comité internacional para definir un lenguaje universal. Fue el precursor de la programación estructurada y tuvo gran influencia posterior.
- **1967**. Aparece Simula, basado en ALGOL, como el primer lenguaje que introduce la idea de clases y objetos. El campo de aplicación principal de este lenguaje era la simulación.
- **1971**. Surge Pascal, basado en ALGOL, como primer lenguaje basado en la programación estructurada. Tuvo gran impacto educativo y sus descendientes son Modula-2/3 y Delphi.
- **1972**. Aparece C, con un menor nivel de abstracción, que aporta una gran flexibilidad y control sobre los recursos de la máquina. Muy relacionado con el entorno UNIX.
- **1973**. Surge SmallTalk como el primer lenguaje orientado a objetos puro.
- **1975**. Aparece PROLOG como el primer lenguaje de programación lógica. Al igual que LISP, se aleja del concepto de la máquina de Von Neumann y se basa en la lógica de primer orden.
- **1983**. Se crea ADA, basado en Pascal, pero más complejo. Permite abordar adecuadamente la programación concurrente y el manejo de excepciones e introduce el concepto de sobrecarga.
- **1983**. Aparece C++ como una evolución de C que mezcla la programación imperativa con la orientada a objetos. Incorpora una biblioteca estándar de clases.
- **1995**. Surge Java como lenguaje de programación orientada a objetos puro con sintaxis de C++. Es independiente de la plataforma, soporta la concurrencia y elimina la aritmética de punteros.
- **1996**. Aparecen los lenguajes script para la programación en entorno web como JavaScript, PHP y ASP. Se caracterizan porque son lenguajes interpretados que se mezclan con el código HTML.
- **2000**. Desarrollado por Microsoft, C# combina las características de C, C++, Java y Visual Basic para la programación en el entorno de desarrollo .NET.