Algoritmos y programas

Algoritmo

Secuencia ordenada de pasos que resuelve un problema concreto.

Características

- Corrección
  (sin errores).
- Precisión
  (ausencia de ambigüedades).
- Repetitividad
  (solución genérica de un problema dado).
- Finitud
  (número finito de órdenes no implica finitud).
- Eficiencia
  (temporal [tiempo necesario] y espacial [memoria utilizada])

Programa

Implementación de un algoritmo en un lenguaje de programación

Conjunto ordenado de instrucciones que se dan al ordenador indicándole las operaciones o tareas que ha de realizar para resolver un problema.

Lenguajes de programación

Una **instrucción** es un conjunto de símbolos que representa una orden para el ordenador: la ejecución de una operación con datos.

Las instrucciones se escriben en un lenguaje de programación:

- Se forman con símbolos tomados de un determinado repertorio (componentes léxicos)
- Se construyen siguiendo unas reglas precisas (sintaxis)

Lenguaje máquina

El único que entiende directamente la CPU del ordenador

✗ Depende del modelo de ordenador

✗ Repertorio de instrucciones reducido (operaciones muy elementales)

✗ Muy difícil programar en él (en binario, con cadenas de ceros y unos)

Lenguaje ensamblador

Equivalente al lenguaje máquina, cada línea de código se traduce en una instrucción para la máquina.

✓ Le asocia mnemónicos a las operaciones que entiende la CPU

✗ Repertorio de instrucciones reducido (operaciones muy elementales)

✗ Programas difíciles de entender

Lenguajes de alto nivel

Permiten que el programador exprese el procesamiento de datos de forma simbólica, sin tener en cuenta los detalles específicos de la máquina.

✓ Independientes del modelo de ordenador

✓ Proporcionan un mayor nivel de abstracción

Ejemplos de lenguajes de programación de alto nivel

FORTRAN (FORmula TRANslation)
© 1957, IBM (John Backus)
Orientado a la resolución de problemas científicos y técnicos

COBOL (COmmon Business Oriented Language)
© 1959, Codasyl (Committee on Data System Languages)
Aplicaciones comerciales de gestión

LISP (LISt Processing)
© 1959, John McCarthy (MIT)
Procesamiento de datos no numéricos (usado en IA)

BASIC (Beginner's All-purpose Symbolic Instruction Code)
© 1964, John Kemeny & Thomas Kurtz (Darmouth College)
Lenguaje interactivo para principiantes

Simula
© 1967, Ole-Johan Dahl & Krysten Nygaard (Noruega)
Primer lenguaje de programación orientada a objetos

Pascal
© 1971, Niklaus Wirth
Lenguaje estructurado diseñado para aprender a programar

C
© 1972, Denis Ritchie (Bell Labs)
Lenguaje pequeño, flexible y eficiente

Smalltalk
© 1972, Alan Kay (Xerox PARC)
Origen de los interfaces WIMP (Windows, Icons, Mouse & Pull-down menus)

PROLOG (PROgramming in Logic)
© 1972, Alain Colmerauer (Universidad de Marsella)
Basado en Lógica (usado en IA)

Ada
© 1980, US Department of Defense
Basado en Pascal, muy usado en aplicaciones militares

C++
© 1983, Bjarne Stroustroup (AT&T Bell Labs)
Extensión de C que permite la programación orientada a objetos

Java
© 1995, Sun Microsystems
Similar a C++, aunque más sencillo de aprender y usar.

C#
© 2000, Microsoft Corporation
Alternativa de Microsoft a Java, muy similar a éste

Clasificación de los lenguajes de programación de alto nivel

- **Lenguajes imperativos**:
  Los programas indican al ordenador de forma inequívoca los pasos a seguir para la resolución de un problema.

  - **Programación estructurada**:
    La estructura del texto del programa debe auxiliarnos para entender la función que realiza: estrategia "divide y vencerás" (la resolución de un problema se divide en tareas y, éstas, en subtareas).

    Ejemplos: C, Pascal, Fortran...

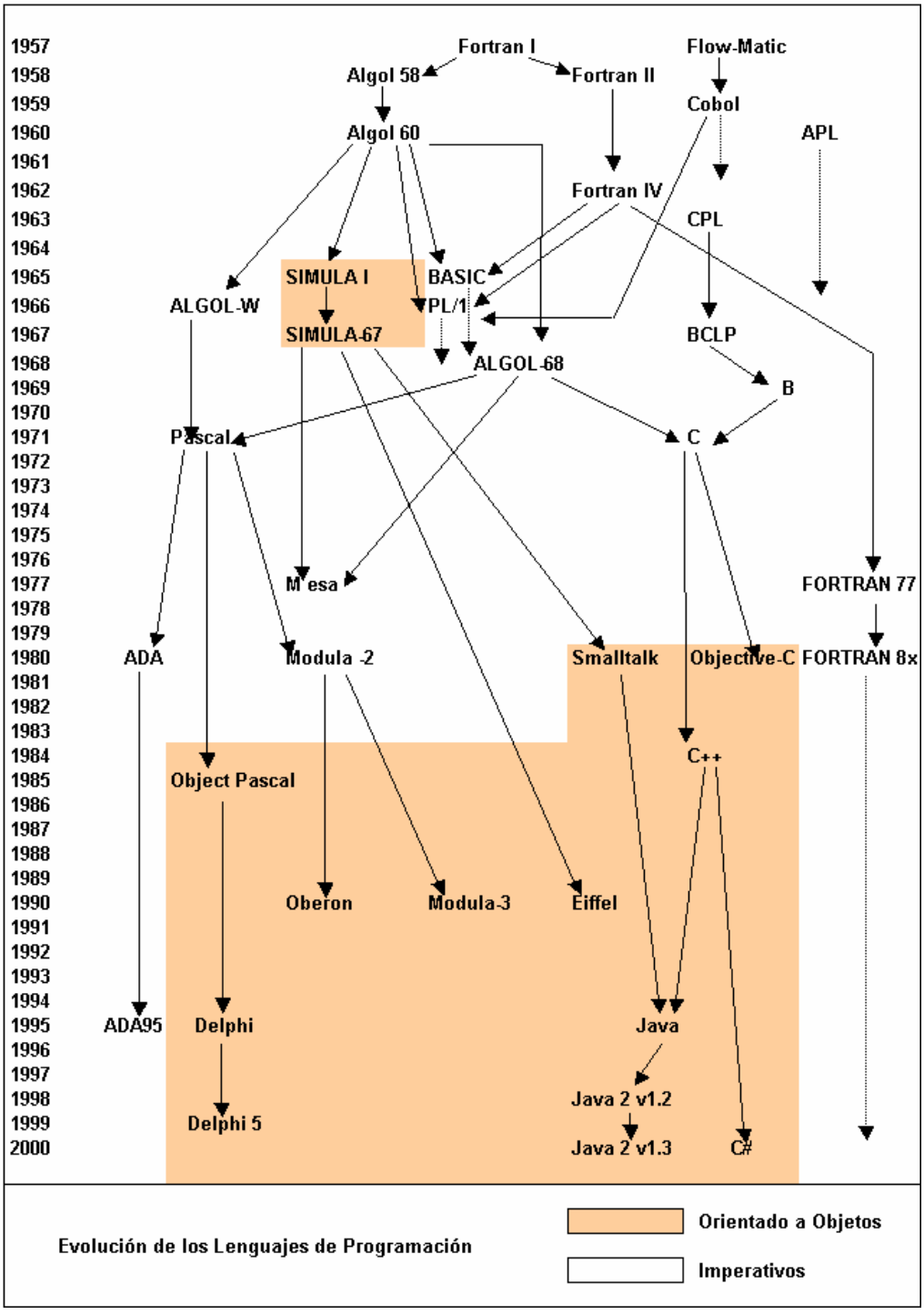
  - P**rogramación orientada a objetos:**
    Estilo de programación que basa la estructura de un programa en módulos deducidos de los tipos de objetos que manipula (en lugar de basarse en las tareas que el sistema debe realizar).

    Ejemplos: Smalltalk, C++, Java, C#...

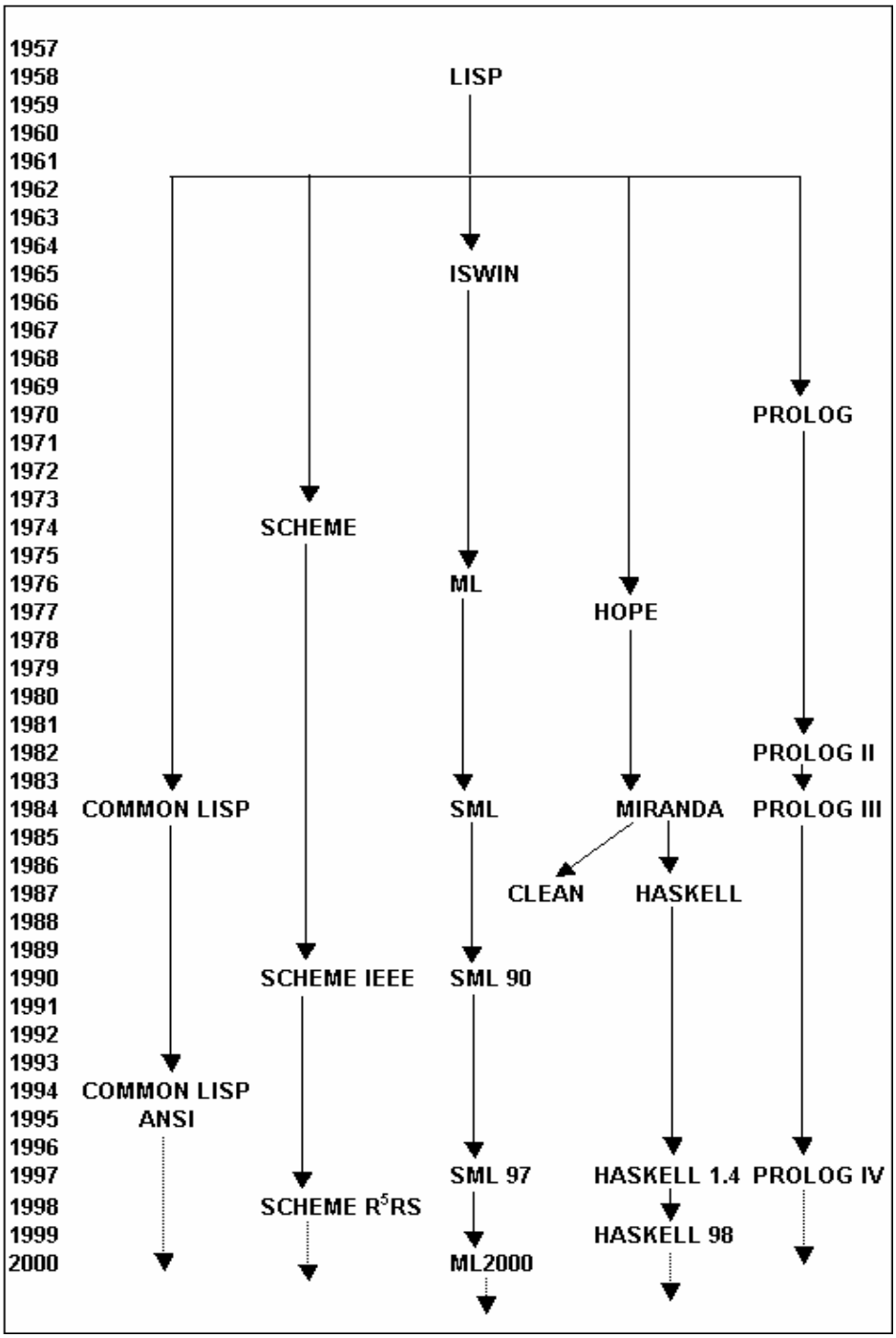
- **Lenguajes declarativos (funcionales y lógicos):**
  Los programas se implementan como conjuntos de funciones (o reglas lógicas) cuya evaluación nos dará el resultado deseado.

  Ejemplos: LISP, PROLOG...

Evolución de los lenguajes de programación: Lenguajes imperativos

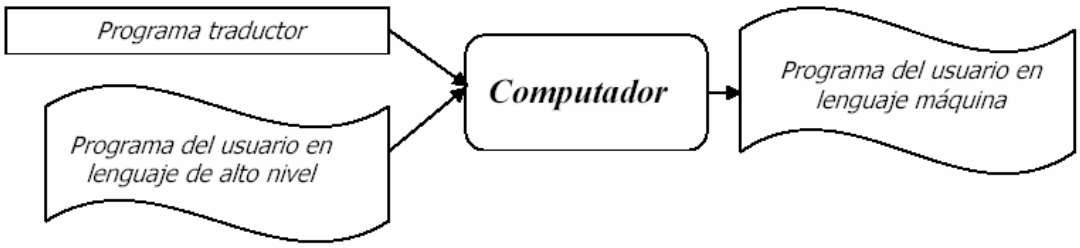


Evolución de los lenguajes de programación: Lenguajes declarativos



Traductores

Los traductores transforman programas escritos en un lenguaje de alto nivel en programas escritos en código máquina:



Tipos de traductores

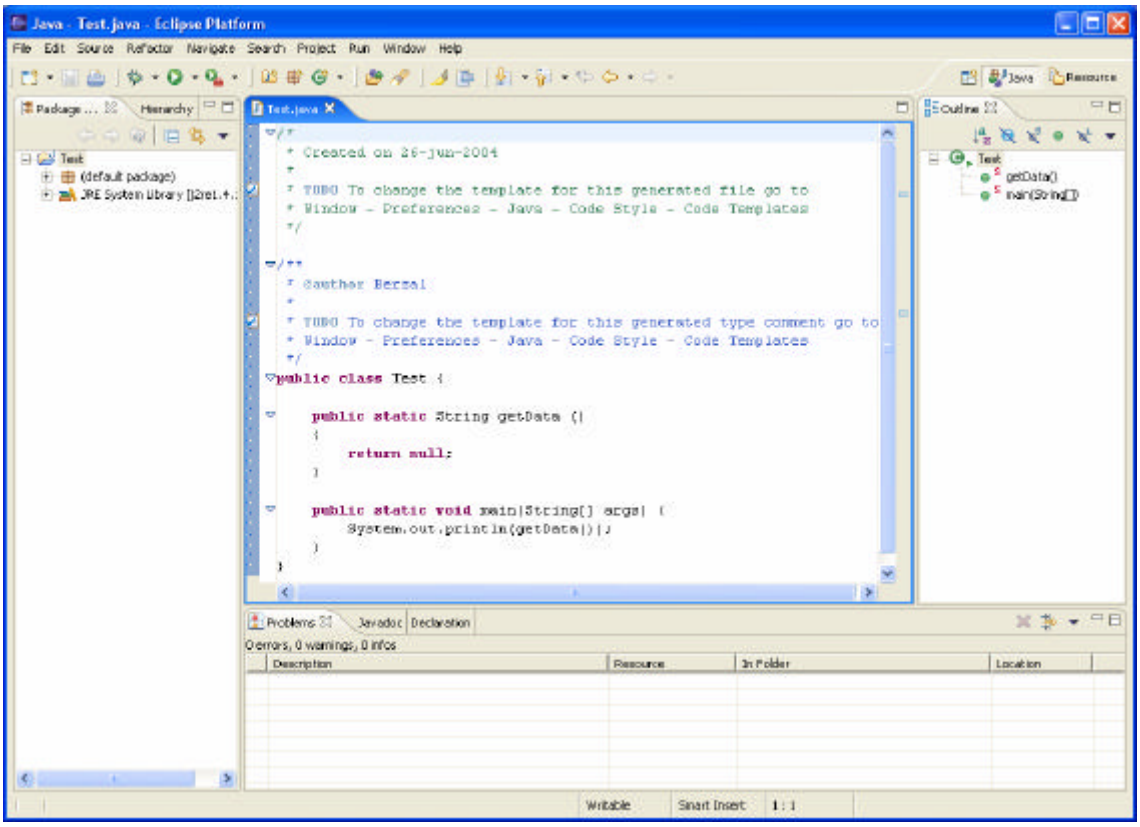
Compiladores
Generan un programa ejecutable a partir del código fuente

Intérpretes
Van analizando, traduciendo y ejecutando las instrucciones del programa una a una. No se traduce una instrucción hasta que la ejecución de la anterior haya finalizado.

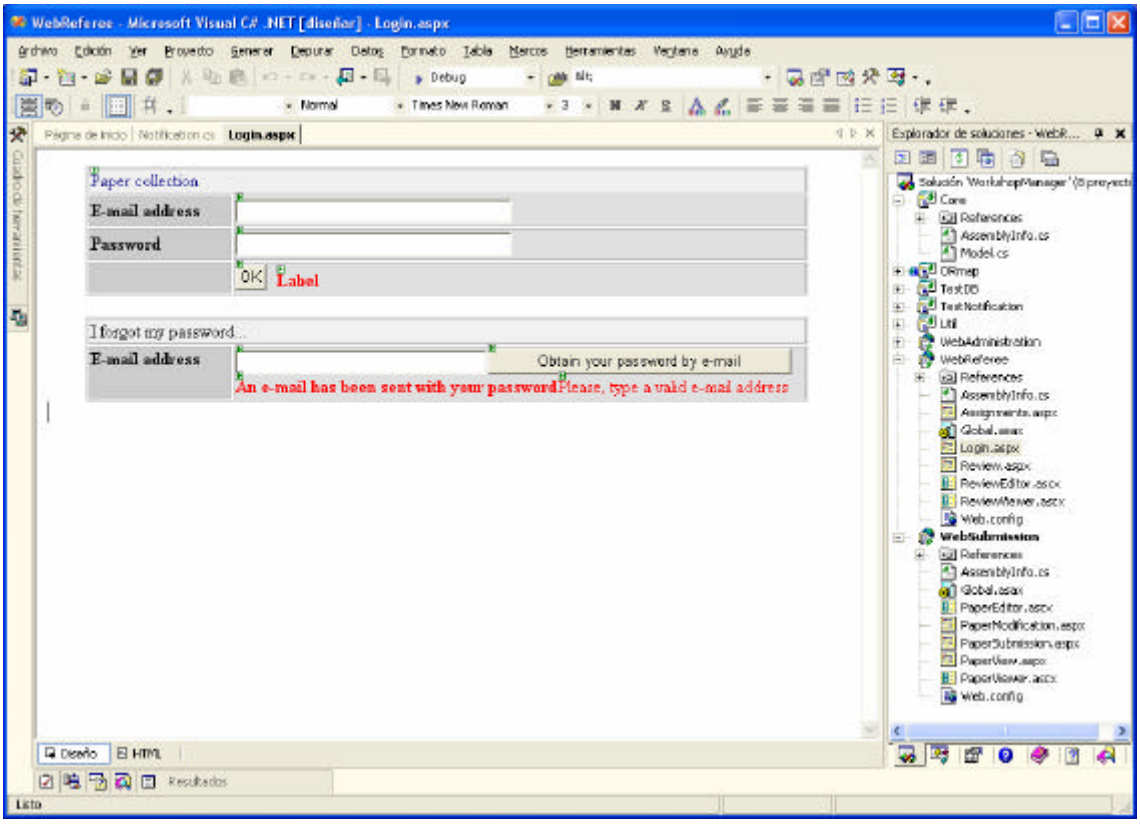
Herramientas de programación

Editores, depuradores, profilers...
↳ IDEs (entornos integrados de desarrollo)

Ejemplos Microsoft Visual Studio .NET
Borland C++Builder/Delphi
Eclipse
…



Eclipse, un IDE para Java (http://www.eclipse.org)



Microsoft Visual Studio .NET, un IDE para la plataforma .NET

Desarrollo de aplicaciones informáticas: Ciclo de vida del software

El ciclo de vida de una aplicación comprende las siguientes etapas:

✓ **Planificación**

Delimitación del ámbito del proyecto, estudio de viabilidad, análisis de riesgos, estimación de costos, planificación temporal y asignación de recursos.

✓ **Análisis** (¿qué?): Elicitación de requisitos.

Descripción clara y completa de qué es lo que se pretende, incluyendo la presentación de los resultados que se desean obtener (formato de las salidas) y la forma en que se va a utilizar la aplicación (interfaz de usuario)

✓ **Diseño** (¿cómo?): Estudio de alternativas

- *Diseño arquitectónico*: Organización de los distintos módulos que compondrán la aplicación (diseño arquitectónico).
- *Diseño detallado*: Definición de los algoritmos necesarios para implementar la aplicación en lenguaje natural, mediante diagramas de flujo o en pseudocódigo [lenguaje algorítmico].

✓ **Implementación**:

Adquisición de componentes, creación de los módulos de la aplicación en un lenguaje de programación e integración de los recursos necesarios para que el sistema funcione.

✓ **Depuración y pruebas**:

Comprobación del funcionamiento de la aplicación

Pruebas de unidad y de integración, pruebas alfa, pruebas beta, test de aceptación.

- *Verificación* (si se está realizando lo que se pretendía)
- *Validación* (si se realiza lo correcto).

✓ **Explotación**: Uso y mantenimiento

- *Mantenimiento correctivo*: Corrección de defectos o errores.
- *Mantenimiento adaptativo*: Adaptación de la aplicación a nuevas circunstancias e inclusión de nuevas prestaciones.