Lenguajes de Programación

Bloque 2. Complemento a Tema 1

1. Historia
2. Paradigmas de programación
3. Fases creación de un programa
4. Traducción: compilación e interpretación

1ª Generación

- Ordenadores sólo entienden lenguaje máquina (instrucción + operando en hexadecimal)
- Cada ordenador tiene su lenguaje propio
- Se hace referencia a las posiciones de memoria por su nombre
- Se busca cambiar valores numéricos por mnemónicos

2ª Generación

- Lenguajes de bajo nivel: lenguaje ensamblador
  - después es necesario traducirlo a lenguaje máquina (usando un assembler-ensamblador)
  - sigue siendo dependiente de la máquina
  - Solución diseñada a muy bajo nivel

3ª Generación

- Lenguajes de alto nivel
  - primitivas del lenguaje son independientes de la máquina
  - 1 instrucción a alto nivel = N instrucciones de lenguaje máquina
  - necesita un proceso de traducción (compilación o interpretación) para pasar a lenguaje máquina
- Independencia de la máquina
  - Con un traductor adecuado un programa en un lenguaje puede llevarse a otra máquina distinta
  - Creación de normas ANSI e ISO para garantizar la compatibilidad

4ª-5ª Generación

- Aplicaciones vs lenguajes de programación
  - Antes: Lenguajes de propósito general Fortran, Basic, C, Pascal, C++,...
  - Ahora entornos ligados a aplicaciones:
    - Visual Basic, Visual C++
    - Paquetes científicos (herederos de Fortran): Maple, Matlab,..
    - Sistemas inteligentes (herramientas de Inteligencia Artificial)
    - Paquetes específicos: Lenguajes simulación
    - Programación web,...
    - Aplicaciones gestión de información: Bases de datos (SQL),...
    - Programación de sistemas: C, awk,Perl,...
- Futuro (5ª generación): ¿paradigma declarativo a bajo nivel?

Paradigmas de programación

- Imperativo
  - Procesamiento secuencial de instrucciones, que actúan sobre variables y tienen E/S
  - Abstracción procedimental
  - Ejemplos: Fortran, Pascal, C,...
- Orientado a Objetos
  - Objetos con operadores se comunican entre sí mediante mensajes para resolver problemas
  - Existen conceptos de herencia y polimorfismo
  - Ejemplos: Smalltalk, Java, *C*++ y Eiffel

Paradigmas de programación

- Funcional
  - Programa = colección de funciones matemáticas (cada una con su dominio e imagen) que pueden interactuar entre ellas y combinarse mediante: condicionales, recursividad y composición funcional
  - Ejemplos: ML, Lisp, Scheme y Haskell
- Lógica o declarativa
  - Especificamos nuestro conocimiento del dominio y las formas que conocemos para manipularlo (cómo resolverlo) en un lenguaje de la lógica; un motor de inferencia resolverá el problema (si puede) y obtendrá las soluciones posibles
  - Ejemplo: Prolog

Otros paradigmas de programación

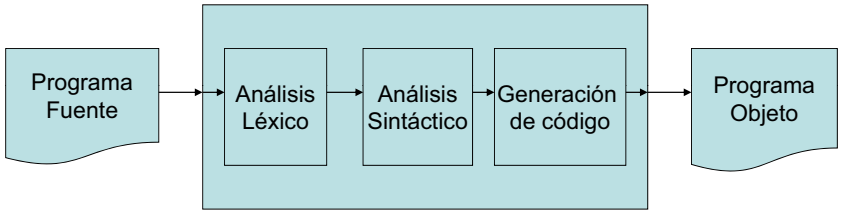
- Programación guiada por eventos
- Programación concurrente
- Programación paralela,...

Proceso de creación de un programa

- El bloque 2 de la asignatura explicará cómo llegar a la generación de un algoritmo para resolver un problema
- El algoritmo después habrá que traducirlo a un programa (en un lenguaje de programación real)
- Aquí, vamos a describir el proceso desde la creación de un programa a alto nivel hasta la generación del programa en lenguaje máquina

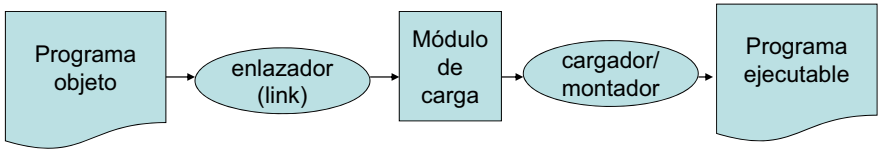
- Etapas:
  1. Traducción
  2. Enlace y Carga

Proceso de traducción de un programa



Proceso de enlace y carga de un programa

- Un programa objeto puede formar parte de una unidad/aplicación mayor



Programa + otros módulos o bibliotecas

En memoria, con posiciones de memoria absolutas

Compilación vs Interpretación

- Compilación
  - fuente → traducción (todas instrucciones) → objeto → ejecutable

  Ejemplo: FORTRAN, C, Pascal,...
- Interpretación
  - fuente → intérprete → instrucciones en LM (para cada instrucción)

  Ejemplo: LISP, Java
- Los lenguajes compilados son más rápidos que los interpretados (pues el código ejecutable se genera en un sólo paso)
- Los lenguajes interpretados son más útiles en la fase de desarrollo. Los compilados cuando la aplicación sólo sufre pequeñas modificaciones