

Tema 1: Introducción

- 1.1 Compiladores y traductores
- 1.2 Características de los lenguajes de programación
- 1.3 Estructura de un compilador
- 1.4 Compiladores de varias pasadas
- 1.5 Entorno software del compilador
- 1.6 Herramientas para la construcción de compiladores
- 1.7 Especificación y diseño de compiladores

1.1 Compiladores y traductores

- Traductores
 - » Lenguaje fuente
 - » Lenguaje objeto
 - » Lenguaje de implementación
- Compiladores
- Intérpretes

Traductores

- *Traducir*
- Lenguaje fuente
- Lenguaje objeto
- Lenguaje de implementación
- Notación T

Compiladores

- *Compilación*
- Lenguaje fuente
 - Lenguaje de alto nivel
- Lenguaje objeto
 - » Lenguaje de bajo nivel
 - » ensamblador, código máquina
- Avisos de error

Intérpretes (I)

- *Interpretar*
- Interpretar-Ejecutar
- La interpretación conlleva
 - » Reconocimiento
 - » Traducción
 - » Ejecución

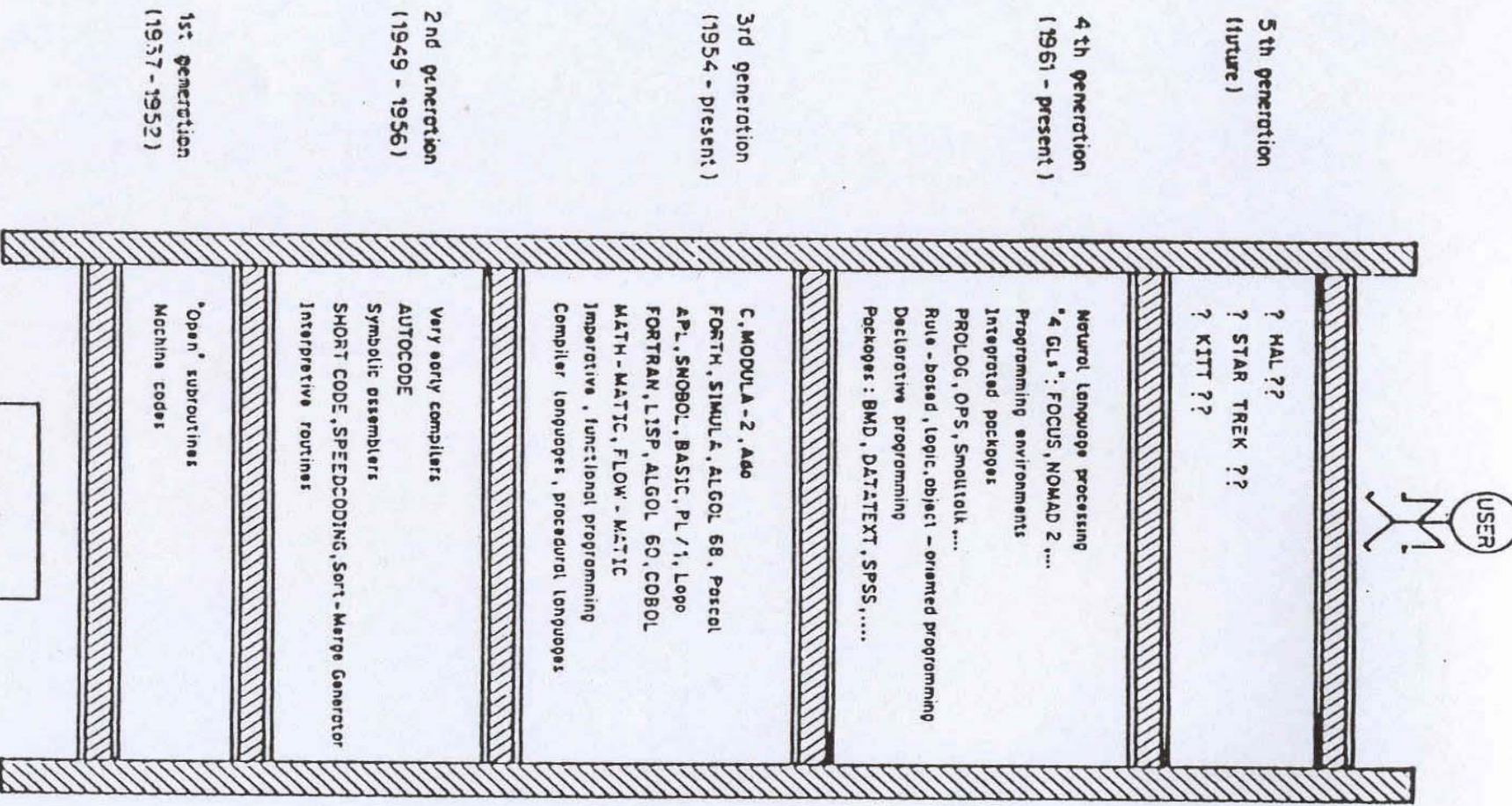
Intérpretes (II)

- Interpretación directa de las instrucciones de alto nivel
- Interpretación en dos fases
 - » Reconocimiento y traducción a código intermedio
 - » Interpretación/ejecución del código intermedio

.2 Características generales de los lenguajes de programación de alto nivel

- Evolución histórica
- Principales características
- Definición de un lenguaje de programación
- Estructura del lenguaje, tipos de datos y estructuras de control
- Clasificación de los lenguajes de programación

Evolución histórica (I)



Evolución histórica (II)

- Lenguajes máquina y lenguajes ensambladores: nivel más bajo de abstracción
- FORTRAN
- COBOL
- ALGOL
 - Incorpora gestión dinámica y Recursividad

Evolución histórica (III)

- **Lenguajes de propósito general**
 - » Orientados a la Programación Estructurada
 - » Pascal → Modula-2 → Ada
 - » Lenguaje C → C++ → Java → C#
- **Lenguajes Especializados**
 - » Lisp: Tratamientos Lineales y arborescentes.
 - » Prolog: Programación Lógica
 - » Php: interfaces web
 - » ...

Evolución histórica (IV)

www.levenez.com/lang/

www.oreilly.com/go/languageposter

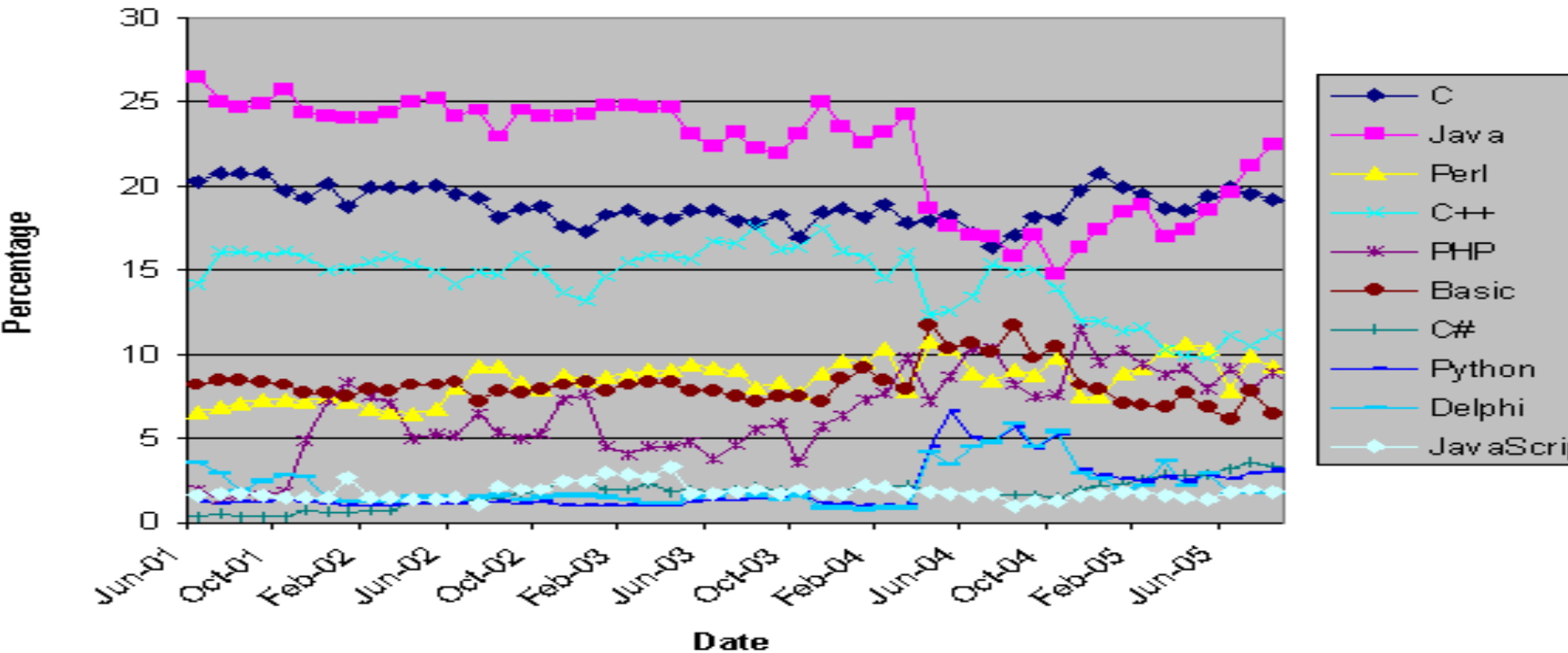
Evolución del uso - estimaciones (I)

<http://www.tiobe.com/>

TIOBE Programming Community Index for September 2005

September Headline: C# seems to be the only .NET language that is going to sta

TPCI Long Term Trends



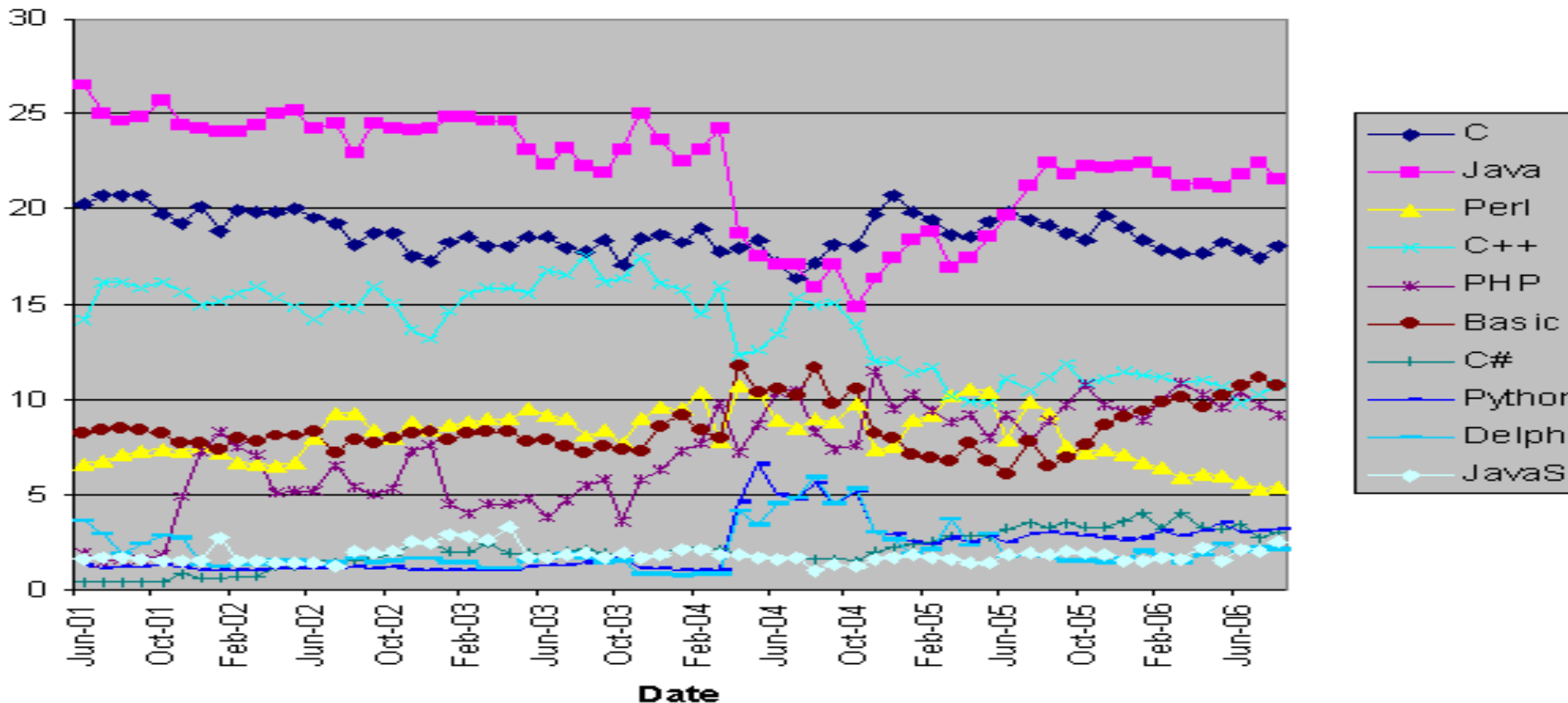
Evolución del uso – estimaciones (II)

<http://www.tiobe.com/>

TIOBE Programming Community Index for September 2006

September Headline: September Headline: Ruby and D are the hot languages of today

TPCI Long Term Trends



Principales características

- Sencillos de aprender y comprender
- Autodocumentados
- Permiten la portabilidad
- Sencillez de mantenimiento y actualización
- Resolución rápida de problemas
- Posibilidad de “debugging”

Definición de un lenguaje de programación (I)

- Incluye la forma (sintaxis) y el significado (semántica) de las instrucciones del lenguaje
- Puede desarrollarse con diferentes niveles de concreción y rigor formal
 - » Usuario del lenguaje
 - » Desarrollador del compilador
- Requisitos: claridad, precisión y abarcar todas las características del lenguaje

Definición de un lenguaje de programación (II)

- Forma/Sintaxis
 - » De las unidades léxicas
 - » De la relación entre las unidades léxicas
- Significado/Semántica
 - » Formal
 - Axiomática
 - Denotacional
 - Operacional
 - » Informal

Estructura del lenguaje

- Facilidades para la estructuración de programas
 - » Estructuración en subprogramas (procedimientos, funciones)
 - » Imbricación de subprogramas
 - » Encapsulación de tipos de datos y operaciones

Tipos de datos

- Tipos de datos básicos
 - » entero, real, carácter, boolean, dirección,...
- Tipos de datos estructurados
 - » Tablas (arrays)
 - » Registros
 - » Conjuntos
 - » Listas

Estructuras de control

- Instrucciones de repetición
 - » while, repeat, loop
- Condicionales
 - » if, case, switch
- Llamadas a subprogramas
- Instrucciones de ruptura de control
 - » exit, goto, continue

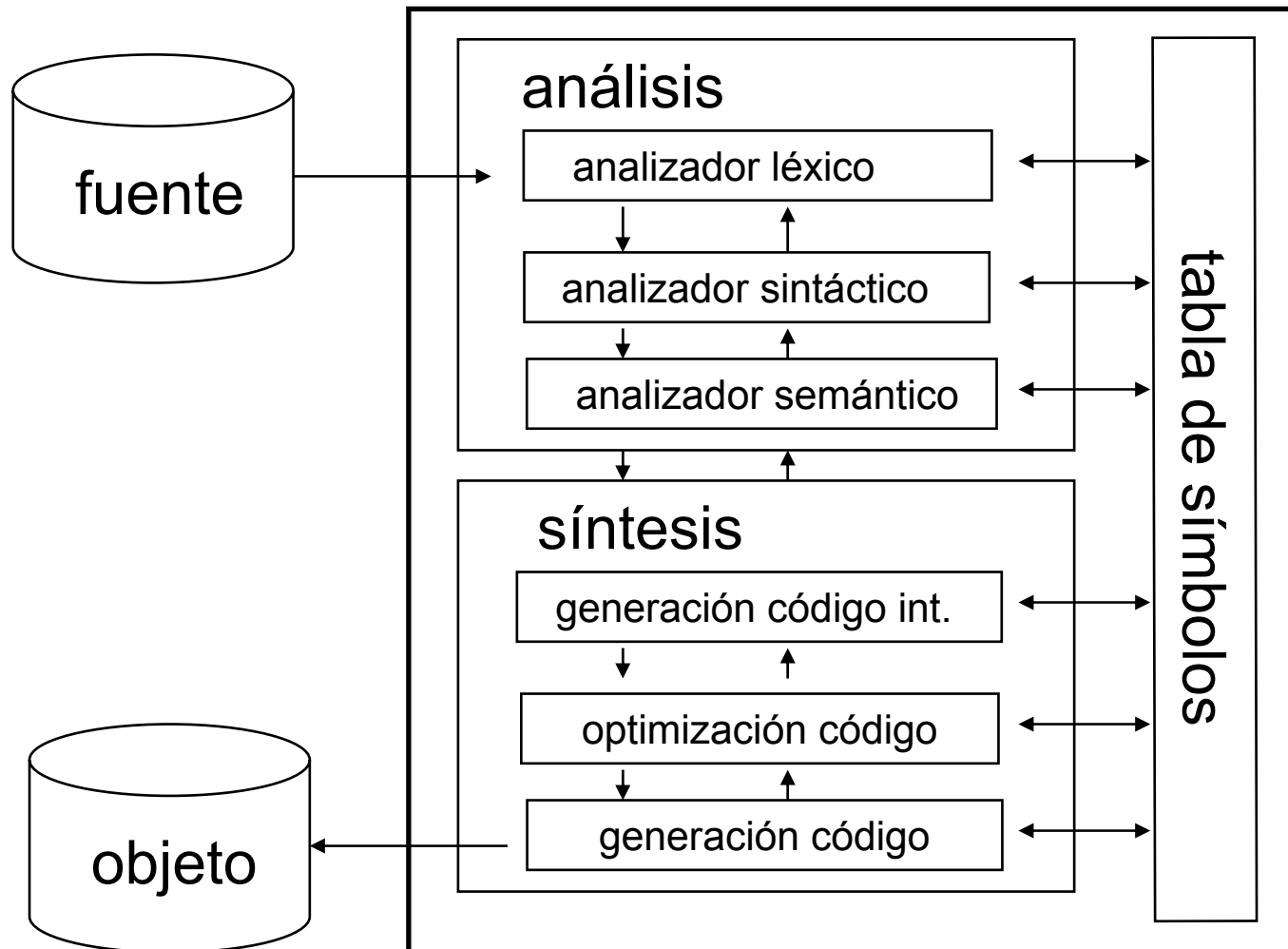
Clasificación de los lenguajes de programación

- Prog. Imperativa
- Lenguajes Funcionales
- Lenguajes Lógicos
- Lenguajes Orientados a Objetos
- Lenguajes Orientados a Eventos

1.3 Estructura de un compilador

- Análisis léxico
- Análisis sintáctico
- Análisis semántico
- Tratamiento de errores
- Manejo de la Tabla de Símbolos
- Generación de código intermedio
- Optimización de código
- Generación de código
- Front-end / Back-end

Estructura de un compilador



Ejemplo

posición := inicial + velocidad * 60

analizador léxico

$id_1 := id_2 + id_3 * 60$

analizador sintáctico

$id_1 := id_2 + id_3 * 60$

analizador semántico

$id_1 := id_2 + id_3 * \text{entareal}(60)$

generador de código intermedio

```
temp1 := entareal(60)
temp2 := id3 * temp1
temp3 := id2 + temp2
idl := temp3
```

optimador de código

```
temp1 := id3 * 60.0
idl := id2 + temp1
```

generador de código

```
MOV id3, R2
MULF #60.0, R2
MOV id2, R1
ADD R2, R1
MOV R1, idl
```

TABLA DE SÍMBOLOS

1	posición	...
2	inicial	...
3	velocidad	...
4		

COMPI-MIN(I)

- Tres instrucciones
 - » lee
 - » escribe
 - » asignación
- Dos tipos de datos
 - » enteros
 - » reales

COMPI-MIN (II)

- Declaración de variables
 - » var lista_de_identificadores: tipo
- Las instrucciones separadas por punto y coma (;)
- (* Los comentarios tienen esta forma *)

COMPI-MIN (ejemplo)

```
(***** mi primer programa *****)  
var hora, minuto, segundo, total : entero;  
lee hora;  
lee minuto;  
lee segundo;  
total:=hora*3600+minuto*60+segundo;  
escribe total; (* escribe resultado *)
```

COMP-INT

```
read hora;  
read minuto;  
read segundo;  
t1:=hora * 3600;  
t2:=minuto*60;  
t3:=t1+t2;  
t4:=t3+segundo;  
total:=t4;  
write total;  
halt;
```

Ejercicio

- Especificar los lexemas (tipos de token) de COMPI-MIN
- Especificar la sintaxis de COMPI-MIN
- Especificar las restricciones semánticas de COMPI-MIN
- Especificar la traducción de los programas escritos en COMPI-MIN a COMP-INT
- Diseñar un programa que analice y traduzca programas escritos en COMPI-MIN

1.4 Compiladores de varias pasadas

- Separa la tarea de compilación en varias pasadas
- Se considera pasada a las escrituras y lecturas sobre ficheros reales
- Es común agrupar varias fases en una pasada, y entrelazar la actividad de estas fases durante la pasada
- Separar la tarea en pasadas tiene ventajas e inconvenientes

1.5 Entorno software del compilador

- Preprocesadores
- Ensambladores
- Linkeditores y cargadores
- Decompiladores

1.6 Herramientas para la construcción de compiladores

- Generación de analizadores y traductores
 - » de analizadores léxicos
 - » de analizadores sintácticos
 - de “evaluadores de ETDSs”
- Generación automática de código

1.7 Especificación y diseño de compiladores

- Diagramas-T
- Bootstrapping (arranque)
- Traducción incremental/interactiva
- Del proyecto UNCOL a la JVM
- Compilación cruzada

Ejercicios

- Se dispone de un compilador de C para PC y se quiere obtener un compilador de Pascal para PC
- Se dispone de un compilador de C para PC y se quiere obtener un intérprete de Pascal para PC
- Se dispone de un intérprete de C para PC y se quiere obtener un compilador de C para PC
- Se dispone de un intérprete de C para PC y se quiere obtener un compilador de Pascal para PC