Procesadores de Lenguajes

Tema 1

Introducción

Tema 1 – Introducción

Sesión 1: Conceptos básicos de compilación e interpretación

Sesión 1: Compiladores e Intérpretes

- Código máquina y lenguajes de alto nivel
- Compiladores v.s. intérpretes
- Esquema y tareas de un compilador:
  - Análisis léxico
  - Análisis sintáctico
  - Análisis semántico
  - Generación de código intermedio
  - Generación de código objeto
  - Optimización de código
- El proceso programación-compilación-ejecución (PCE)
- Tipos de errores en el proceso PCE

Código máquina y lenguajes de alto nivel (1)

- Hoy por hoy, salvo en prototipos de investigación, los ordenadores sólo procesan el alfabeto eléctrico de los 5 (V) y de los 0 (V), es decir, de un voltaje que denota el valor “1” o “0”, respectivamente, reconociendo sólo el lenguaje generado por estos dos dígitos (el **código máquina**).
- Con ese alfabeto pueden resolverse todos los problemas computables (mediante lo que se llama una **máquina de *Turing***)
  - Inconveniente: hay que encontrar una definición de las entradas, el proceso y las salidas del algoritmo que soluciona cada problema en términos del alfabeto binario.
  - ¿Cómo soslayar este inconveniente?

Código máquina y lenguajes de alto nivel (2)

- Lo más práctico es:
  - Resolver los problemas con otro alfabeto, menos restrictivo, y un lenguaje más genérico que el código máquina (el **lenguaje de alto nivel**), pero con reglas más estrictas, que permitan hacer programas más fácilmente
  - Encontrar la forma de traducir el lenguaje de alto nivel a código máquina de forma automática
- Los programas que traducen un lenguaje de alto nivel a código máquina son los **compiladores** y los **intérpretes**
- Ventaja añadida:
  - El programa codificado en lenguaje de alto nivel puede compilarse o interpretarse, a priori, en cualquier máquina
  - El problema de traducir el lenguaje de alto nivel al código máquina queda resuelto "para siempre" (para ese lenguaje de alto nivel y para esa máquina)

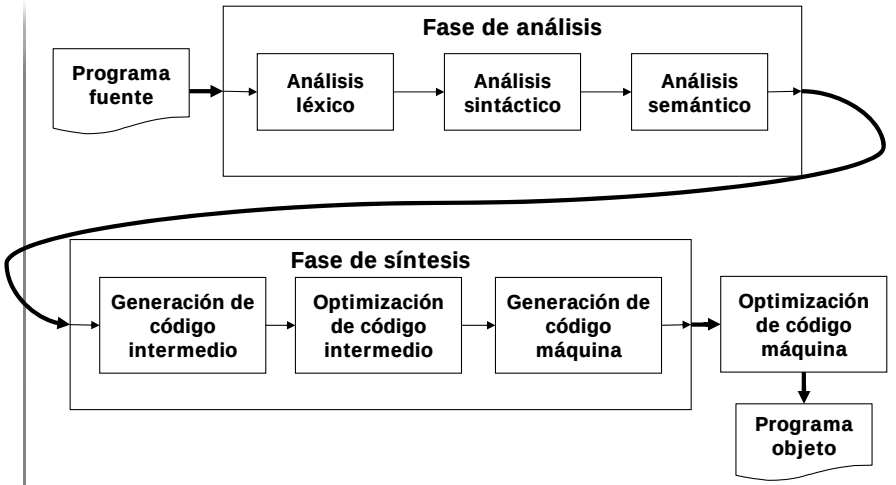
Compiladores v.s. intérpretes (1)

- Definiciones:
  - Compilador:
    - Es un programa que (secuencialmente):
      - comprueba que un programa escrito en un lenguaje de alto nivel es gramaticalmente correcto
      - traduce dicho programa a código máquina
  - Intérprete:
    - Es un programa que, **a un mismo tiempo**:
      - comprueba que un programa escrito en un lenguaje de alto nivel es gramaticalmente correcto
      - traduce dicho programa a código máquina
      - ejecuta el programa traducido a código objeto

Compiladores v.s. intérpretes (2)

- Comparando ambos tipos de traductores:
  - El compilador:
    - Reduce el tiempo de depuración del programa (no hace dos tareas a la vez, sino sólo una, con lo que tarda la mitad de tiempo)
    - Almacena el código objeto, de forma que no hay que volver a compilar para volver a ejecutar
  - El intérprete:
    - Permite la ejecución de programas aún no completados
    - No suele requerir una definición del tipo de los datos, sino que se infieren automáticamente, con lo que se ahorra cierto tiempo en la programación
    - Permiten toda una serie de operaciones complejas, ya predefinidas, que en lenguajes compilados tardan mucho en programarse (equiparación de patrones, por ejemplo)

Esquema genérico de un compilador



Descripción de las tareas de un compilador

- Análisis léxico:
  - Identificación de palabras y símbolos del lenguaje.
  - Comprobación de las reglas léxicas.
- Análisis sintáctico:
  - Comprobación de las reglas sintácticas.
- Análisis semántico:
  - Comprobación de las reglas semánticas (variables no declaradas, comprobación de tipos, etc.).
  - Interpretación de las órdenes.
- Generación de código:
  - Intermedio: Traduce las instrucciones del lenguaje de alto nivel a un lenguaje próximo al lenguaje máquina, pero aún independiente de la máquina en la que se va a ejecutar el programa
  - Objeto: Traduce las instrucciones del código intermedio al lenguaje máquina. Se pierde la independencia de la máquina
- Optimización de código: Análisis del código para mejorarlo (en velocidad, en espacio de memoria)

Tareas de un compilador: ejemplo (1)

- **CÓDIGO FUENTE:**
  - Secuencia de caracteres ASCII.
- **Análisis LÉXICO:**
  - id: identificador.
  - ent: constante entera.
  - op: operador.
  - punto_coma.
- **CÓDIGO FUENTE:**

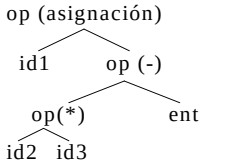
  [lim, largo, alto : REAL;]

  lim := largo * alto -1;
- **Lista de TOKENS:**
  - id(lim)
  - op(asignación)
  - id(largo)
  - op(*)
  - id(alto)
  - op(-)
  - ent(1)
  - punto_coma

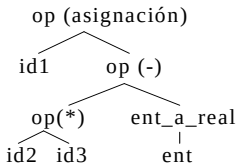
Tareas de un compilador: ejemplo (2)

- **An. SINTÁCTICO:**
  - $S \rightarrow E$ **;**
  - $E \rightarrow F$ op F
  - $F \rightarrow$ **id** | **ent** | **real** | E
- **An. SEMÁNTICO:**
  - Comprobación de tipos.
  - Conversión implícita.

♦ An. SINTÁCTICO.



♦ An. SEMÁNTICO.



Tareas de un compilador: ejemplo (3)

- **G. C. INTERMEDIO:**
  - Instrucciones de una máquina abstracta.
- **OPTIMIZACIÓN C. I.:**
  - Minimización de accesos a memoria.
  - Evaluación de expresiones constantes.
- **G. C. INTERMEDIO:**
  - t1 := id2 * id3
  - t2 := ent_a_real(1)
  - t3 := t1 - t2
  - id1 := t3
- **OPTIMIZACIÓN C. I.:**
  - t1 := id2 * id3
  - id1 := t1 - 1.0

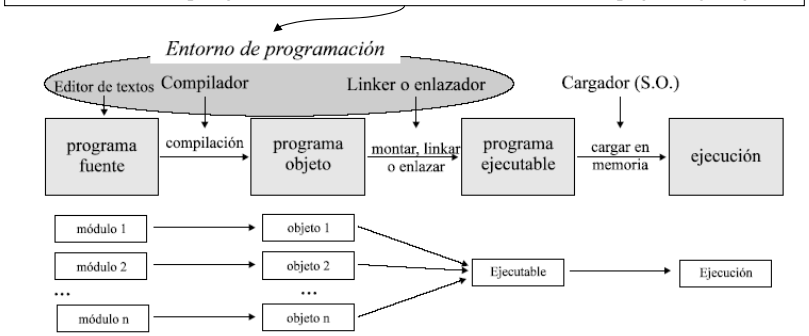
Tareas de un compilador: ejemplo (4)

- **G. C. FINAL:**
  - Secuencia de instrucciones en código ejecutable o ensamblador IEEE Std. 694, por ejemplo (más portable).
- **OPTIMIZACIÓN C. F.:**
  - Minimización de accesos a memoria.
  - Optimización del uso de registros.

- **G. C. FINAL:**
  - LD .R1, /id2
  - LD .R2, /id3
  - MUL .R1, .R2
  - ST .R1, /t1
  - SUB .R1, 1.0
  - ST .R1, /id1
- **OPTIMIZACIÓN C. F.:**
  - LD .R1, /id2
  - LD .R2, /id3
  - MUL .R1, .R2
  - SUB .R1, 1.0
  - ST .R1, /id1

El proceso programación-compilación-ejecución (PCE)

Turbo Pascal [http://bdn.borland.com/article/20803], por ejemplo



Tipos de errores en el proceso PCE

- Errores de compilación:
  - Los producidos en la fase de compilación o interpretación de un programa, es decir, cuando no se cumplen las reglas léxicas, sintácticas o semánticas.
- Errores de ejecución:
  - Los producidos durante la ejecución del programa. Estos mensajes de error no son producidos por el compilador, sino por una porción de código que el compilador añade al programa.
- Errores lógicos:
  - Cuando el programa no da ningún error, pero los resultados no son los esperados. Puede ser porque el algoritmo estaba mal diseñado o porque el algoritmo ha sido incorrectamente implementado.

Tema 1 – Introducción

Un ejemplo: Pascal

Ejemplo: Introducción informal a Pascal

- Reglas gramaticales
  - En los lenguajes naturales y en los lenguajes formales
  - Notación
    - BNF
    - Diagramas sintácticos
- Pascal: Alfabeto y símbolos léxicos
  - Semántica de los símbolos léxicos
  - Sintaxis de los símbolos léxicos invariables: comentarios, operadores, signos de puntuación y palabras reservadas
- Pascal: Sintaxis global
  - Programas en Pascal: estructura básica y vista global
  - Elementos de un programa en Pascal: bloques y declaraciones
  - Esquema de un programa en Pascal

Reglas gramaticales – en el lenguaje humano

- Todo lenguaje humano tiene sus propias reglas:
  - ortográficas ("Antes de una *p* nunca va una *n*")
  - morfológicas ("No se dice *hacido*, sino *hecho*")
  - sintácticas ("La oración se compone de sujeto y predicado")
  - semánticas ("En general, el verbo *ser* detalla una característica permanente del sujeto, mientras *estar* detalla una característica transitoria")

Reglas gramaticales – en los lenguajes formales

- Asimismo, todo lenguaje formal consta de unas definiciones, denominadas **reglas sintácticas** o **producciones**, que especifican qué construcciones son válidas y cuáles no en el lenguaje.
- Las reglas sintácticas pueden contener dos tipos de elementos:
  - Elementos Terminales ($\in$ Léxico)
  - Elementos No Terminales, que son construcciones intermedias de la gramática

Notación de reglas: BNF

- Notación BNF (Backus-Naur Form)
  - Es una de las primeras notaciones que se empezaron a utilizar para especificar las reglas gramaticales de los lenguajes de programación
  - Esquema general de una regla:

    <elemento no terminal> ***::=*** *Definición1 | Definición2 | ...*
    - Los elementos terminales, que pertenecen al léxico, se escriben tal cual
    - Los elementos no terminales se escriben entre los símbolos '<' y '>'
    - El símbolo '|' indica diferentes alternativas de definición

Notación de reglas en BNF: Ejemplo 1

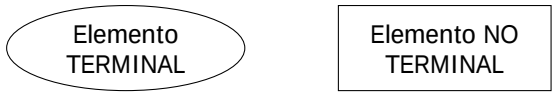
- "La oración (O) se compone de sujeto (Suj) y de predicado (Pred)"
  <O> ::= <Suj> <Pred>
- "El sujeto puede ser un sintagma nominal (SNom), un nombre propio (NP), un pronombre (Pron) o, incluso, no aparecer (λ), dándose por sobreentendido"
  - Ejemplos:
    - **El vecino de arriba** vino tarde | **Juan** vino tarde | **Él** vino tarde | Vino tarde
  - Regla:
    - <Suj> ::= <SNom> | <NP> | <Pron> | λ
- "Un sintagma nominal está compuesto o bien por un determinante (Det), un nombre común (NC) y un sintagma adjetivo (SAdj), o bien por un determinante, un nombre común y un sintagma preposicional (SPrep), o bien por un artículo (Art), un adjetivo (Adj) y un nombre común"
  - Ejemplos:
    - Ese hombre viejo | Ese hombre de edad avanzada | El viejo hombre
  - Regla:
    - <SNom> ::= <Det> <NC> <SAdj> | <Det> <NC> <SPrep> | <Art> <Adj> <NC>
- "Los artículos, en español, son: el, la, lo, los, las, un, uno, una, unos y unas"
  <Art> ::= el | la | lo | los | las | un | uno | una | unos | unas

Notación de reglas en BNF: Ejemplo 2

- **Ejemplo:** Descripción sintáctica, en notación BNF, de una expresión matemática del tipo: *4*(3+1)*
  - <expresión> ::= <número> | (<expresión>) | <expresión><operador><expresión>
  - <operador> ::= + | - | * | /
  - <número> ::= <dígito> | <número><dígito>
  - <dígito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

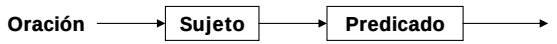
Notación de reglas: Diagramas sintácticos

- Una alternativa más visual para la descripción de la gramática de un lenguaje:

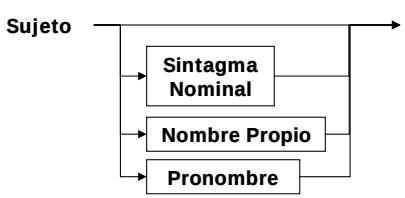


Diagramas sintácticos: Ejemplo 1

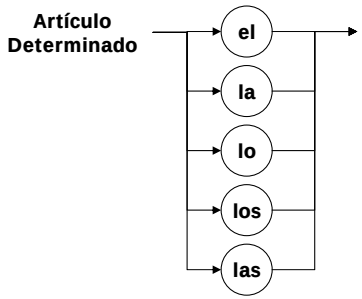
"La oración (O) se compone de sujeto (Suj) y de predicado (Pred)"



"El sujeto puede ser un sintagma nominal (SNom), un nombre propio (NP), un pronombre (Pron) o, incluso, no aparecer (λ), dándose por sobreentendido"

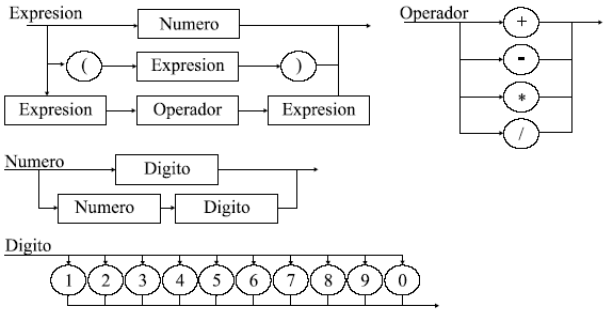


"Los artículos determinados, en español, son: el, la, lo, los, las, un, uno, una, unos y unas"



Diagramas sintácticos: Ejemplo 2

- **Ejemplo:** Descripción sintáctica, mediante diagramas sintácticos, de una expresión matemática del tipo: *4*(3+1)*



Pascal: alfabeto y léxico

- El alfabeto de Pascal
- El léxico de Pascal
  - Semántica de los símbolos léxicos
  - Sintaxis de los símbolos léxicos invariables:
    - Comentarios
    - Operadores
    - Signos de puntuación
    - Palabras clave (o reservadas)

Pascal: alfabeto y léxico

- Alfabeto
  - Los caracteres del alfabeto inglés (no están permitidos los caracteres acentuados, ni la 'ñ' fuera de los comentarios, por ejemplo), los dígitos decimales, los operadores matemáticos, lógicos y de comparación, así como los signos de puntuación.
- Léxico
  - Conjunto de símbolos que se pueden usar en un lenguaje. Aparte de los operadores y los signos de puntuación, pertenecen al léxico:
    - Los comentarios
    - Las palabras clave o reservadas
    - Las constantes
    - Los identificadores
    - Las instrucciones

Pascal: elementos léxicos – semántica (1)

- Conjunto de símbolos invariables de Pascal:
  - Comentarios:
    - Texto que acompaña a una sección del programa, documentándolo y clarificando el código (aunque son también útiles a la hora de buscar errores de compilación que se resisten).
  - Operadores:
    - Símbolos que representarán operaciones entre variables y constantes.
  - Signos de puntuación:
    - Como en el lenguaje natural, son símbolos que delimitan otros símbolos y las sentencias del programa (la coma, el punto, el punto y coma, los dos puntos, los paréntesis, etc.)
  - Palabras clave o reservadas:
    - Son símbolos indivisibles, con una semántica predefinida en el lenguaje, y que no pueden usarse para ningún otro propósito que aquél para el que se han definido en el lenguaje (excepto en comentarios)

Pascal: elementos léxicos – semántica (2)

- Conjunto de símbolos de Pascal construidos por el usuario:
  - Identificadores:
    - Nombres simbólicos que se darán a ciertos elementos de programación (por ejemplo: nombres de constantes, de tipos, de variables, de programas, etc.).
    - Deben ser convenientemente declarados, para ser reconocidos por el compilador.
  - Constantes:
    - Datos que no cambiarán su valor a lo largo del programa.
  - Instrucciones:
    - Símbolos compuestos que representarán estructuras de procesamiento y de definición de elementos de programación.

Pascal: elementos léxicos invariables – sintaxis (1.a)

- Comentarios:
  - <comentario> ::= (* <cualquier cosa> *) | { <cualquier cosa> }
- Operadores:
  - <operador> ::= <op_matemático> | <op_comparación> | <op_lógico> | := | ↑
  - <op_matemático> ::= + | - | * | / | div | mod | shr | shl | in
  - <op_comparación> ::= < | > | <= | >= | = | <>
  - <op_lógico> ::= not | and | or

    (* independientemente de mayúsculas/minúsculas *)

- Signos de puntuación:
  - <puntuación> ::= . | , | ; | : | ^ | ' | " | (|) | [|] | { | }

Pascal: elementos léxicos invariables – sintaxis (1.b)

- Palabras clave (o reservadas) [Grogono, 1996]

AND	ARRAY	BEGIN		CASE		CONST
DIV	DO	DOWNTO		ELSE		END
FILE	FOR	FUNCTION		GOTO		IF
IN	LABEL	MOD		NIL		NOT
OF	OR	PACKED		PROCEDURE		PROGRAM
RECORD	REPEAT	SET	SHL	SHR	THEN	TO
TYPE	UNTIL	VAR		WHILE		WITH

Bibliografía

- Brassard, G., Bratley, P. (1997) Fundamentos de Algoritmia. Madrid:Prentice Hall.
- Grogono, P. (1996) Programación en Pascal. Wilmington, Delaware (EE.UU.):Addison-Wesley Iberoamericana.
- Joyanes Aguilar, L. (2006) Programación en Pascal (4ª edición). McGraw-Hill/Interamericana de España, S.A.
- Leestma, S., Nyhoff, L. (1999) Programación en Pascal (4ª edición). Madrid:Prentice Hall.
- Valls Ferrán, J.M., Camacho Fernández, D. (2004) Programación estructurada y algoritmos en Pascal. Madrid:PEARSON-Prentice Hall.