Resumen Conceptos y paradigmas de los lenguajes de programación

1 - Sintaxis y semántica

Lenguaje de programación: notación formal para describir algoritmos a ser ejecutados en una computadora. Compuesto por sintaxis y semántica.

Sintaxis: conjunto de reglas que definen como componer letras, dígitos y otros caracteres para formar los programas.
Características:

- Debe ayudar al programador a escribir programas correctos sintácticamente
- Establece reglas que sirven para que el programador se comunique con el procesador.
- Debe contemplar soluciones a características tales como:
    - Legibilidad
    - Verificabilidad
    - Traducción
    - Falta de ambigüedad

Elementos:

- Alfabeto o conjunto de caracteres
- Identificadores
- Operadores
- Palabra clave y palabra reservada
- Comentarios y uso de blancos

Estructura de la sintaxis:

- Vocabulario o words (se construyen a partir del alfabeto)
- Expresiones
- Sentencias

Reglas léxicas: conjunto de reglas para formar las “word”, a partir de los caracteres del alfabeto.
Reglas sintácticas: conjunto de reglas que definen cómo formar las “expresiones” y “sentencias”

Tipos de sintaxis:

- Abstracta: se refiere a la estructura
- Concreta: se refiere a la parte léxica
- Pragmática: se refiere al uso práctico

Formas de definir la sintaxis:

- Lenguaje natural: ej Fortran
- Utilizando la gramática libre de contexto, definida por Backus y Naun: BNF o EBNF
- Diagramas sintácticos: equivalente a BNF pero mucho más intuitivos

BNF:

- Notación formal para describir la sintaxis
- Es un metalenguaje
- Utiliza meta-símbolos: < > ::== |
- Define las reglas por medio de producciones. Ej: < digito > ::== 0|1|2|3|4|5|6|7|8|9
- Gramática: conjunto de reglas finitas que definen un conjunto infinito de posibles sentencias válidas en el lenguaje. Formada por una 4-tupla
    - G = (N,T,S,P) = (No terminales, terminales, símbolo distinguido de la gramática que pertenece a N, conjunto de producciones

  Una gramática es ambigua si una sentencia puede derivarse de más de una forma.

EBNF: BNF extendido incorpora:

- [ ]:elemento optativo puede o no estar
- (|): selección de una alternativa
- { }: repetición
- * 0 o más veces
- + una o más veces

Semántica: conjunto de reglas para dar significado a los programas sintácticamente válidos. Describe el significado de los símbolos, palabras y frases de un lenguaje ya se lenguaje natural o informático

La definición de la sintaxis y la semántica de un lenguaje de programación proporcionan mecanismos para que una persona o una computadora pueda decir:

- Si el programa es válido.
- Si lo es, qué significa.

Tipos de semántica

- Semántica estática:
    - No está relacionado con el significado del programa está relacionado con las formas válidas.
    - Se las llama así porque el análisis para el chequeo puede hacerse en compilación
- Semántica dinámica: es la que describe el efecto de ejecutar diferentes construcciones en el lenguaje de programación. Su efecto se describe durante la ejecución del programa.

Formas de describir la semántica:

- Semántica axiomática:
    - Considera al programa como “una máquina de estados”
    - La notación empleada es el “cálculo de predicados”
    - Se desarrolló para probar la corrección de programas.
    - Los constructores de los lenguajes de programación se formalizan describiendo como su ejecución provoca un cambio de estado.
- Semántica denotacional:

    - Se basa en teoría de funciones recursivas
    - Se diferencia de la axiomática por la forma en la que describe los estados, la axiomática los describe a través de predicados, la denotacional a través de funciones.
    - Se define una correspondencia entre los constructores sintácticos y sus significados
- Semántica operacional:
    - El significado de un programa se describe mediante otro lenguaje de bajo nivel implementado sobre una máquina abstracta.
    - Los cambios se producen en el estado de la máquina cuando se ejecuta una sentencia del lenguaje de programación definen su significado
    - Es un método informal
    - Es el más utilizado en libros de texto
    - PL/1 fue el primero que la utilizo

Alternativas de traducción de alto nivel hacia bajo nivel:
- Interpretación
- Compilación

Intérprete:
- Lee
- Analiza
- Decodifica
- Ejecuta una a una las sentencias del programa

Por cada posible acción hay un subprograma que ejecuta esa acción. La interpretación se realiza llamando a estos subprogramas en la secuencia adecuada

Compilación: los programas escritos en alto nivel se traducen a una versión en lenguaje de máquina antes de ser ejecutados.
Pasos:
- Compilación a Assembler
- Ensamblado a código reubicable
- Linkeditado
- Cargado en la memoria

2 - Semántica operacional

Entidades: Variable, Rutina, Sentencia

Variable → Atributos:
- Nombre
- Tipo
- Área de memoria
- Valor

Rutina → Atributos:

- Nombre
- Parámetros formales
- Parámetros reales

Sentencia → Atributos: acción asociada

Concepto de ligadura (binding): programas trabajan con entidades → atributos → tienen que establecerse → ligadura: es la asociación entre la entidad y el atributo

Ligadura estática: se establece antes de la ejecución y no se puede cambiar. El término estático se refiere al momento del binding y a su estabilidad.
Ligadura dinámica: se establece en momento de ejecución y puede cambiarse de acuerdo a una regla específica del lenguaje. (Excepción constante no puede cambiarse)

Variable: una variable es una abstracción de una celda elemental de memoria principal identificada por una dirección. El contenido de una celda es la representación codificada de un valor. Sus atributos son:

- Nombre: string de caracteres que se utiliza para referenciar a la variable. (identificador)
- Alcance: es el rango de instrucciones en el que se conoce el nombre.
- Tipo: define los posibles valores y operaciones a realizar
- L-Value: es el lugar de memoria asociado con la variable (tiempo de vida)
- R-Value: es el valor codificado almacenado en la ubicación de la variable

Alcance Estático:

- Llamado alcance léxico
- Define el alcance en términos del a estructura léxica del programa.
- Puede ligarse estáticamente a una declaración (explícita o implícita) examinando el texto del programa sin necesidad de ejecutarlo
- Es la que adoptan la mayoría de los lenguajes

Alcance Dinámico:

- Define el alcance del nombre de la variable en términos de la ejecución del programa.
- Cada declaración de variable extiende su efecto sobre todas las instrucciones ejecutadas posteriormente, hasta que una nueva declaración para una variable con el mismo nombre es encontrado durante la ejecución
- APL, Lisp (original), Afnix (llamado Aleph hasta el 2003), Tcl (Tool Command Language), Perl

Conceptos asociados al alcance:

- Local: son todas las referencias que se han creado dentro del programa o subprograma.
- No Local. son todas las referencias que se utilizan dentro del subprograma pero no han sido creadas en el.
- Global: son todas las referencias creadas en el programa principal

Tipo. Momentos

- Estático: el tipo se liga en compilación y no puede ser cambiado
    - el chequeo de tipos también será estático
    - Estático explícito: la ligadura se establece mediante una declaración: ej int x,y
    - Estático implícito: la ligadura se establece por reglas
    - Estático Inferido: el tipo de una expresión se deduce por el tipo de sus componentes
- Dinámico: el tipo se liga en ejecución y puede cambiarse
    - Más flexible programación genérica
    - Más costoso en ejecución: mantenimiento de descriptores
    - Variables polimórficas
    - Chequeo dinámico
    - Menor legibilidad

L-Value. Momentos - Alocación:

- Estática: sensible a la historia
- Dinámica
    - Automática: cuando aparece la declaración
    - Explícita: a través de algún constructor
- Persistente: su tiempo de vida no depende de la ejecución (base de datos)

R-Value: Momentos

- Dinámico: se puede modificar el valor
- Constantes: se congela el valor

Sobrecarga: 1 nombre → distintas entidades
Alias: distintos nombres → 1 entidad (distintos nombres mismo l-value)

Unidades de un programa: Procedimientos y funciones

- Nombre: string de caracteres utilizado para invocar la rutina
- Alcance: rango de instrucciones donde se conoce su nombre
- Tipo: el encabezado de la rutina define el tipo de los parámetros y tipo de valor de retorno (si lo hay)
- L-Value: es el lugar de memoria que se almacena el cuerpo de la rutina
- R-Value: la llamada a la rutina causa la ejecución del código, eso constituye su r-valor:
    - estático: caso más usual
    - dinámica: variables de tipo rutina. Se implementan a través de punteros a rutinas.

Comunicación entre rutinas: parámetros

- Parámetros formales: los que aparecen en la definición del a rutina.
- Parámetros reales: los que aparecen en la invocación de la rutina. (dato o rutina)
- Métodos de pasaje de parámetros:

    - Posicional
    - Por nombre

3 - Esquemas de ejecución

Representación en ejecución:

- Segmento de código: instrucciones de la unidad. Se almacena en la memoria de instrucción. Contenido fijo
- Registro de activación: datos locales de la unidad. Se almacena en la memoria de datos. Contenido cambiante

Estructuras de ejecución:

- Estático: espacio fijo
    - El espacio necesario para la ejecución se deduce del código
    - Todo los requerimientos de memoria necesarios se conocen antes de la ejecución.
    - La alocación puede hacerse estáticamente
    - No puede haber recursión.
- Basado en pila
    - El espacio se deduce del código
    - Programas más potentes cuyos requerimientos de memoria no pueden calcularse en traducción.
    - La memoria a utilizarse es predecible y sigue una disciplina last-in-first-out.
    - Las variables se alocan automáticamente y se desalocan cuando su alcance termina.
    - Se utiliza una estructura de pila para modelizarlo.
    - Una pila no es parte de la semántica del lenguaje, es parte de nuestro modelo semántico
- Dinámico
    - Lenguajes con impredecible uso de memoria
    - Los datos son alocados dinámicamente sólo cuando se los necesita durante la ejecución.
    - No pueden modelizarse con una pila, el programador puede crear objetos de datos en cualquier punto arbitrario durante la ejecución del programa
    - Los datos se alocan en la zona de memoria heap