Introducción

Contenidos

1 Modelos de computación
2 Paradigmas de programación
3 Historia
4 Otras cuestiones

Introducción

Modelos de computación

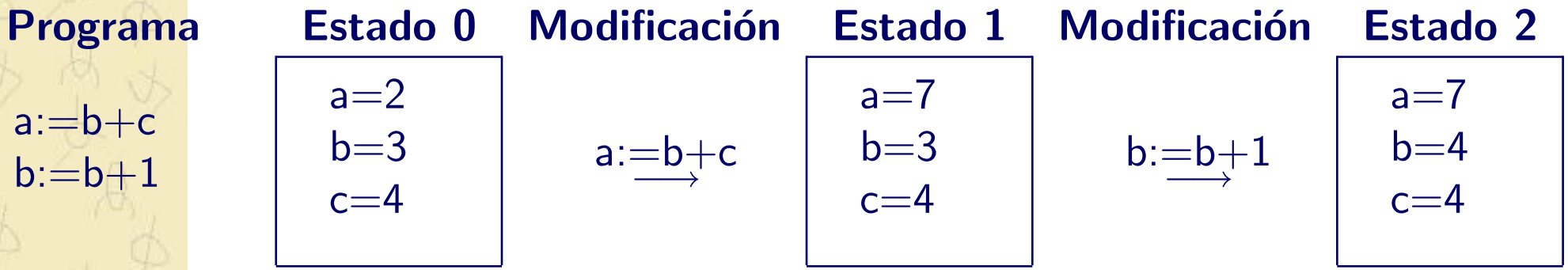
- Modelos de computación
2 Paradigmas de programación
3 Historia
4 Otras cuestiones

Lenguajes de programación

- **Modelo computacional:** colección de valores y operaciones
- **Tipos de modelos computacionales:** imperativo, funcional y lógico
- **Computación:** aplicación de una secuencia de operaciones a un valor para obtener otro valor
- **Programa:** especificación de una computación
- **Lenguaje de programación:** notación para escribir programas
- **Sintaxis de un lenguaje de programación:** estructura o forma de los programas
- **Semántica de un lenguaje de programación:** relaciones entre un programa y un modelo de computación
- **Pragmática de un lenguaje de programación:** grado de éxito con el que un programa cumple sus objetivos tanto en su fidelidad con el modelo de computación subyacente como su utilidad para los programadores

Modelo de computación imperativo

- También denominado procedural
- Conjunto de valores que definen un estado y la operación de asignación de la modificación de estado
- Un estado es un conjunto de pares nombre-valor de las constantes y variables
- Un programa es una secuencia de asignaciones
- Una computación es una secuencia de estados
- Ejemplo:



Modelo de computación funcional

- Conjunto de valores, funciones y las operaciones de aplicación de función y composición de función
- Las funciones pueden tomar como argumentos otras funciones y devolverlas como resultados
- Un programa es una colección de definiciones de funciones
- Una computación es una evaluación (aplicación) de una expresión (función con sus argumentos)
- Ejemplo

Programa	Computación	Resultado
(defun member (elemento lista) (cond ((null lista) nil) ((eq elemento (car lista)) lista) (t (member elemento (cdr lista)))))	`(member 'a '(b a c))`	`(a c)`

Modelo de computación lógico

- Conjunto de valores, definiciones de relaciones e inferencias lógicas
- Un programa es una definición de relaciones
- Una computación es una prueba (secuencia de inferencias lógicas)
- Ejemplo:

Programa

```
member(A,[],[]) :- !.
member(A,[A|X],[A|X]) :- !.
member(A,[B|X],Y) :-
  member(A,X,Y).
```

Computación

```
?- member(a,[b,a,c],X)
  X=[a,c]
```

Prueba

```
member(a,[b,a,c],[a,c])
   3. A=a, B=b, X=[a,c], Y=[a,c]
member(a,[a,c],[a,c])
   2. A=a, X=[c]
```

Computabilidad

- Los tres modelos son equivalentes

  cualquier problema que tenga una solución en un modelo es resoluble en cada uno de los otros dos

- Se denominan modelos universales de computación
- Se han planteado otros modelos que también son equivalentes
- Estos tres modelos están mezclados en los lenguajes de programación
  - ⋆ Java es imperativo, pero incorpora elementos funcionales y lógicos
  - ⋆ CommonLisp es funcional, pero permite realizar programación imperativa y lógica
  - ⋆ Prolog es lógico, pero admite determinadas secuencias imperativas

Introducción

Paradigmas de programación

1 Modelos de computación
- Paradigmas de programación
3 Historia
4 Otras cuestiones

Paradigmas de programación

- Programación imperativa
- Programación declarativa
- Programación orientada a objetos
- Programación basada en módulos
- Otros paradigmas

Programación imperativa

- Modela la arquitectura de la máquina de von Neumann
- La computación consiste en la ejecución paso a paso de algoritmos (secuencias condicionales o repetitivas de instrucciones) que modifican los contenidos de variables (espacios de memoria)
- Tipos:
  - ⋆ **Lenguajes procedurales** (Pascal, C): utilizan subprogramas como unidades de modularización para definir pasos de computación
  - ⋆ **Lenguajes orientados a objetos** (Smalltalk, C++, Java): encapsulan en objetos tanto los datos como las operaciones sobre los mismos

Ejemplo de programación imperativa

```
Function member (e: real; a: lista-reales): lista-reales;
    begin
        if (a = nil)
        then member := nil
        else while ((a^ .siguiente <> nil) and
                        (e <> a^ .elemento)) do
                a := a^ .siguiente;
            if e = a^ .elemento
            then member := a
            else member := nil
    end;
```

Programación declarativa

- Utiliza bloques de construcción como las funciones, la recursión o la equiparación de patrones, para especificar más la solución que su cálculo de bajo nivel
- Tipos:
  - ⋆ **Lenguajes funcionales** (Common Lisp, Scheme, Haskell): utilizan funciones libres de efectos secundarios como bloques primitivos de construcción de programas. Estas funciones pueden aplicarse, construirse y pasarse como argumentos a otras funciones
  - ⋆ **Lenguajes lógicos** (Prolog): calculan resultados utilizando reglas e inferencia lógicas

Ejemplo de programación declarativa

- Programación lógica (Prolog)

```
member(A,[],[]) :- !.
member(A,[A|X],[A|X]) :- !.
member(A,[B|X],Y) :-
    member(A,X,Y).
```

- Programación funcional (CommonLisp y Haskell)

```
(defun member (elemento lista)
   (cond ((null lista) nil)
         ((eq elemento (car lista)) lista)
         (t (member elemento (cdr lista)))))
```

```
fact 0 = 1
fact n = n * fact (n - 1)
```

Programación orientada a objetos

- Define la computación como la interacción entre objetos autónomos
- Mediante la utilización de objetos, la programación se convierte en simulación
- Los objetos son instancias de clases que se organizan en jerarquías de herencia
- Los objetos se comportan independientemente
- Utilizan la selección de las operaciones en tiempo de ejecución cuando interpretan los mensajes provenientes de otros objetos
- Ejemplos:
  - ⋆ Smalltalk, Eiffel, y Java son lenguajes orientados a objetos
  - ⋆ Muchos lenguajes actuales, como C++, Ada-95, o Common Lisp también dan soporte a este tipo de programación

Ejemplo de programación orientada a objetos

- Clase: lista
- Atributos: elemento, siguiente
- Métodos: primero, siguiente, member, ...
- Ejecución de member: mandar un mensaje `member` a una instancia de la clase `lista` con el argumento *elemento a buscar*

Programación basada en módulos

- Reconoce agrupaciones de variables, procedimientos y tipos como unidades estáticas de modularidad de programa
- El interface a un módulo (o paquete) especifica y exporta una serie de servicios generales
- Los detalles de implementación se ocultan en una unidad de programa compilada separadamente
- Ejemplos:
  - ⋆ Modula-2, Ada y C++ soportan este estilo de programación que puede ser combinada con cualquier otro tipo de paradigma

Otros paradigmas de programación

- Programación visual
- Programación basada en restricciones
- Programación basada en tipos abstractos de datos
- Programación genérica
- Programación concurrente
- Programación basada en componentes distribuidos
- Programación basada en agentes
- Programación literaria
- Programación basada en patrones de diseño
- Programación basada en guiones
- Programación automática (programación genética)

Introducción

Historia

1 Modelos de computación
2 Paradigmas de programación
• Historia
4 Otras cuestiones

Historia de los lenguajes de programación

- Charles Babbage
- Zuse, Plankalkül (1945)
  - ⋆ tipos de datos: bit, escalares (con bits), matrices, registros recursivos
  - ⋆ no se descubrió hasta los 70
- 1940-50: código máquina, direcciones absolutas, muy poca memoria
- John Mauchly (1949): creo un lenguaje para el ordenador BINAC (luego UNIVAC), interpretado
- John Backus (1954): extendió el lenguaje máquina para operaciones en coma flotante para el IBM 701
- FORTRAN (IBM FORmula TRANslation system) se introdujo en el IBM 704 en 1954

Historia de algunos lenguajes

Lenguaje	Años	Orígenes	Lenguaje predecesor	Propósito y características
FORTRAN	1954-57-58-	J. Backus (IBM)		Computación numérica, 1er lenguaje alto nivel, compilación independiente
LISP	1956-59-	J. McCarthy	FLPL (FORTRAN)	Computación simbólica, 1er lenguaje funcional, procesamiento de listas, garbage collection
Algol 60	1958-60	Comité (USA Europa)	FORTRAN	Computación numérica, estructura de bloque, estructuras de control, recursión
COBOL	1959-60-	Comité		Procedimientos de negocio, registros y E/S
APL	1956-60	K. Iverson		Procesamiento de vectores
SNOBOL	1962-66	R. Griswold (Bell Labs)		Procesamiento de cadenas

Historia (cont. I)

Lenguaje	Años	Orígenes	Lenguaje predecesor	Propósito y características
PL/1	1963-1965	Comité (IBM)	FORTRAN Algol 60 COBOL	Propósito general, tipo de datos puntero, concurrencia, gestión de excepciones
BASIC	1964-	J. Kemeny, T. Kurtz	FORTRAN	Educativo
Simula	1967	O.J. Dahl, K. Nygaard	Algol 60	1er lenguaje OO y de simulación, corutinas, clases, herencia
Algol 68	1963-68	Comité	Algol 60	Propósito general, ortogonalidad, tipos definidos por usuario, matrices flexibles
Pascal	1971	N. Wirth	Algol 60	Educativo, propósito general, pequeño y simple
Prolog	1972	A. Colmerauer		Inferencia lógica (IA)
C	1972	D. Ritchie (Bell Labs)	B, Algol 68	Programación de sistemas
Scheme	1975	G. Steele G. Sussman	LISP	Educativo, simple, funcional
Modula-2	1977	N. Wirth	Pascal	Programación de sistemas, módulos, corutinas

Historia (cont. II)

Lenguaje	Años	Orígenes	Lenguaje predecesor	Propósito y características
Smalltalk	1971-80	A. Kay A. Goldberg	Simula (LISP)	Computación personal, OO puro, pionero en GUI
Ada	1979-83-95	J. Ichbiah	Pascal etc.	Sistemas embebidos, paquetes, concurrencia
C++	1984-	B. Stroustrup (Bell Labs)	C, Simula	Programación de sistemas, multi-paradigma, grande, complejo
Haskell	1989	Comité(?)	ML, Scheme, etc.	Programación simbólica, síntesis de programas, funcional
Tcl/Tk	1988	J.K. Ousterhout	shell scripts	Lenguaje de *scripting*
Perl	1990	L. Wall	shell scripts	Lenguaje de *scripting*
Python	1991	G. vanRossum	shell scripts	Lenguaje de *scripting*
Delphi	1994	BORLAND	Pascal	Propósito general, modular & OO
Java	1995-	SUN Microsystems	C++, Smalltalk	Computación basada en red, Educativo, OO simplificada

Historia. Primeros lenguajes

- FORTRAN
  - ⋆ Backus en IBM, 1957, 18 personas-año para desarrollar el primer compilador
  - ⋆ Metas y contribuciones: problemas numéricos, compilador muy eficiente, compilación separada
  - ⋆ Muchas revisiones: FORTRAN II, IV, 66, 77, 90
- ALGOL
  - ⋆ Comité, 1960
  - ⋆ Metas y contribuciones: numérico, estructura de bloque, recursión, elegante, muy influyente
  - ⋆ Padre del ALGOL W, ALGOL 68, Pascal, Modula, Ada, etc.
- COBOL
  - ⋆ Comité, 1960
  - ⋆ Metas y contribuciones: procesamiento de datos de negocio, registros
  - ⋆ Varias revisiones

Primeras escisiones

- LISP
  - ⋆ McCarthy en MIT, 1962
  - ⋆ Metas y contribuciones: programación funcional, procesamiento de listas, manipulación de símbolos
  - ⋆ Padre de Scheme y CommonLisp
- APL
  - ⋆ Iverson en Harvard, IBM, 1960 (para la notación)
  - ⋆ Metas y contribuciones: lenguaje de cálculo de matrices
- SNOBOL 4
  - ⋆ Griswold en Bell Labs, 1966
  - ⋆ Metas y contribuciones: procesamiento de cadenas de caracteres a través de equiparación de patrones
  - ⋆ Padre de ICON

Historia. Consolidación

- PL/I
  - ⋆ Comité en IBM, 1967
  - ⋆ Combinación de FORTRAN, COBOL, ALGOL 60, pero no integrados
  - ⋆ Multi-propósito, incluye punteros, registros, excepciones
  - ⋆ Se considera un fallo
- ALGOL 68
  - ⋆ Comité
  - ⋆ Elementos ortogonales
  - ⋆ Elegante, pero difícil de entender
- Simula 67
  - ⋆ Precursor de los lenguajes orientados a objetos
  - ⋆ Diseñado para la simulación
  - ⋆ Incluye corutinas

Historia. Consolidación (I)

- Pascal
  - ⋆ Wirth, ETH, 1971
  - ⋆ Diseñado como lenguaje docente
  - ⋆ Permite programación estructurada y es elegante
  - ⋆ Tuvo más éxito que el esperado

FORTRAN

- Primer lenguaje ampliamente aceptado
- Introdujo la facilidad de programación, eficiencia de los programas hechos a mano, y la reducción de errores de codificación
- Énfasis en el cálculo numérico en máquinas con poca memoria y procesadores lentos
- Primitivas del lenguaje
  - ⋆ subrutinas (no se hacía compilación separada)
  - ⋆ formato de E/S
  - ⋆ condicionales de tipo IF (sin operadores relacionales en el juego de caracteres)
  - ⋆ repeticiones con el DO

Historia del FORTRAN

- FORTRAN II (1958): compilación independiente de las subrutinas
- FORTRAN IV (1962-68)
  - ⋆ declaraciones explícitas de tipos
  - ⋆ IF lógico
  - ⋆ subprogramas como parámetros de otros subprogramas
- FORTRAN 77
  - ⋆ cadenas de caracteres
  - ⋆ control lógico de bucles (while)
  - ⋆ la cláusula ELSE para el IF

FORTRAN 90

- Operaciones con matrices pre-definidas
- Matrices dinámicas
- Registros (llamados tipos derivados)
- Instrucción CASE
- Salidas de bucles prematuras
- Recursión en subprogramas
- Módulos

LISP. John McCarthy (1958)

- *List Processing Language*
- Énfasis en cálculo simbólico
- Gestión dinámica de memoria para listas
- Primitivas del lenguaje
  - ⋆ expresiones condicionales
  - ⋆ recursión
  - ⋆ gestión dinámica de memoria
  - ⋆ los programas pueden ser también datos
- Otros lenguajes relacionados: Scheme, Common Lisp, ML

Scheme. Guy L. Steele & Gerard J. Sussman (1984)

- Desarrollado en MIT a partir del Lisp
- Conjunto básico de primitivas
- Utilizado incluso para cursos introductorios de programación

ALGOL

- A mediados de los 50 hubo una proliferación de lenguajes y ACM formó un comité en 1957
- Objetivos:
  - ⋆ sintaxis cercana a la notación matemática
  - ⋆ debía ser usable en publicaciones
  - ⋆ debía ser compilable
- ALGOL 58
  - ⋆ tipos de datos: coma flotante implícita y el resto debían declararse explícitamente
  - ⋆ identificadores de cualquier longitud
  - ⋆ matrices con número arbitrario de dimensiones
  - ⋆ se podían declarar explícitamente los límites inferiores de las matrices
- ALGOL 60
  - ⋆ parámetros se podían pasar por valor y por nombre
  - ⋆ procedimientos recursivos

Introducción

Otras cuestiones

1 Modelos de computación
2 Paradigmas de programación
3 Historia
• Otras cuestiones

Requisitos de un lenguaje de programación

- Universal
- Natural (expresivo)
- Implementable
- Eficiente (escribir, compilar, ejecutar)
- Robusto
- Mantenible

Abstracciones

- Abstracciones de datos
  - ⋆ tipos básicos de datos: enteros, reales, booleanos, caracteres, punteros
  - ⋆ estructurados: matrices, registros
  - ⋆ unidades: soporte a tipos abstractos de datos, módulos, paquetes, clases
- Abstracciones de control
  - ⋆ básicas: asignación, ir-a, secuencia
  - ⋆ estructuradas: if-then-else, bucles, procedimientos (funciones)
  - ⋆ unidades: unidades de compilación separada, módulos, paquetes, tareas concurrentes