El Lisp (o LISP) es una familia de lenguajes de programación de computadora de tipo multiparadigma con una larga historia y una sintaxis completamente entre paréntesis.

Especificado originalmente en 1958 por John McCarthy y sus colaboradores en el Instituto Tecnológico de Massachusetts, el Lisp es el segundo más viejo lenguaje de programación de alto nivel de extenso uso hoy en día; solamente el FORTRAN es más viejo.

Al igual que el FORTRAN, el Lisp ha cambiado mucho desde sus comienzos, y han existido un gran número de dialectos en su historia. Hoy, los dialectos Lisp de propósito general más ampliamente conocidos son el Common Lisp y el Scheme.

El Lisp fue creado originalmente como una notación matemática práctica para los programas de computadora, basada en el cálculo lambda de Alonzo Church. Se convirtió rápidamente en el lenguaje de programación favorito en la investigación de la inteligencia artificial (AI). Como uno de los primeros lenguajes de programación, el Lisp fue pionero en muchas ideas en ciencias de la computación, incluyendo las estructuras de datos de árbol, el manejo de almacenamiento automático, tipos dinámicos, y el compilador auto contenido.

El nombre LISP deriva del «LISt Processing» (Proceso de LIStas). Las listas encadenadas son una de las estructuras de datos importantes del Lisp, y el código fuente del Lisp en sí mismo está compuesto de listas. Como resultado, los programas de Lisp pueden manipular el código fuente como una estructura de datos, dando lugar a los macro sistemas que permiten a los programadores crear una nueva sintaxis de lenguajes de programación de dominio específico empotrados en el Lisp.

La intercambiabilidad del código y los datos también da a Lisp su instantáneamente reconocible sintaxis. Todo el código del programa es escrito como expresiones S, o listas entre paréntesis. Una llamada de función o una forma sintáctica es escrita como una lista, con la función o el nombre del operador en primer lugar, y los argumentos a continuación; por ejemplo, una función f que toma tres argumentos puede ser llamada usando `(f x y z)`.

El Lisp fue inventado por John McCarthy en 1958 mientras estaba en el Instituto Tecnológico

de Massachusetts (MIT). McCarthy publicó su diseño en 1960 en un artículo de Communications of the ACM titulado «funciones recursivas de expresiones simbólicas y su cómputo por la máquina, parte I» (la «parte II» nunca fue publicada). Allí mostró que con algunos operadores simples y una notación para las funciones, uno puede construir un lenguaje turing completo para los algoritmos.

Desde 1955 ó 1956, el Information Processing Language fue el primer lenguaje de AI, y ya había incluido muchos de los conceptos, tales como proceso por lista y recursión, que vinieron a ser usados en el Lisp.

La notación original de McCarthy usaba «expresiones M» en corchetes que serían traducidas a expresiones S. Como un ejemplo, la expresión M `car[cons[A,B]]` es equivalente a la expresión S `(car (cons A B))`. Una vez el Lisp fue implementado, los programadores rápidamente eligieron usar expresiones S, y las expresiones M fueron abandonadas. las expresiones M emergieron otra vez con los intentos efímeros del MLISP de Horace Enea y el CGOL de Vaughan Pratt.

El Lisp fue implementado primero por Steve Russel en un computador IBM 704. Russell había leído el artículo de McCarthy, y se dio cuenta (para la sorpresa de McCarthy) que la función *eval* del Lisp podía ser implementada en código de máquina. El resultado fue un intérprete de Lisp funcional que podía ser usado para correr programas Lisp, o más correctamente, «evaluar expresiones Lisp».

Dos rutinas de lenguaje ensamblador para el IBM 704 se convirtieron en las operaciones primitivas para descomponer listas: car (contenido del registro de dirección) y cdr (contenido del registro del decremento). Los dialectos del Lisp todavía usan el car y cdr (pronunciado /?k?r/ y /?k?d?r/) para las operaciones que retornan el primer elemento y el resto de la lista respectivamente.

El primer compilador completo de Lisp, escrito en Lisp, fue implementado en 1962 por Tim Hart y Mike Levin en el MIT. Este compilador introdujo el modelo Lisp de compilación incremental, en el cual las funciones compiladas e interpretadas pueden entremezclarse

libremente. El lenguaje en los memos de Hart y Levin es mucho más cercano al estilo moderno del Lisp que el anterior código de McCarthy.

Genealogía y variantes

Sobre su historia de cincuenta años, el Lisp ha producido muchas variaciones en el tema base de un lenguaje de expresión S. Por otra parte, cada dialecto dado puede tener varias implementaciones, por ejemplo, hay más de una docena de implementaciones del Common Lisp.

Las diferencias entre los dialectos pueden ser muy visibles, por ejemplo, el Common Lisp y el Scheme usan diferentes palabras clave para definir funciones. Dentro de un dialecto que está estandarizado, sin embargo, las implementaciones conformadas soportan el mismo lenguaje base, pero con diferentes extensiones y bibliotecas.

Relación con la inteligencia artificial

Desde su inicio, el Lisp estaba estrechamente relacionado con la comunidad de investigación de la inteligencia artificial, especialmente en sistemas PDP-10. El Lisp fue usado como la implementación del lenguaje de programación Micro Planner que fue la fundación para el famoso sistema de AI SHRDLU. En los años 1970, a medida que la investigación del AI engendró descendientes comerciales, el desempeño de los sistemas Lisp existentes se convirtió en un problema creciente.

El Lisp era un sistema difícil de implementar con las técnicas de compilador y hardware común de los años 1970. Las rutinas de recolección de basura, desarrolladas por el entonces estudiante graduado del MIT, Daniel Edwards, hicieron práctico correr Lisp en sistemas de computación de propósito general, pero la eficacia todavía seguía siendo un problema. Esto llevó a la creación de las máquinas Lisp: hardware dedicado para correr ambientes y programas Lisp. Avances tanto en el hardware de computadora como en la tecnología de compiladores pronto hicieron obsoletas a las máquinas de Lisp, en detrimento del mercado del Lisp.

Durante los años 1980 y 1990, fue hecho un gran esfuerzo para unificar los numerosos dialectos del Lisp en un solo lenguaje (más notablemente, InterLisp, Maclisp, ZetaLisp, MetaLisp, y Franz Lisp). El nuevo lenguaje, Common Lisp, fue esencialmente un subconjunto compatible de los dialectos que reemplazó. En 1994, la ANSI publicó el estándar del Common Lisp, «ANSI X3.226-1994 Information Technology Programming Language Common Lisp». En aquel momento el mercado mundial para el Lisp era mucho más pequeño de lo que es hoy.

Desde el año 2000

Habiendo declinado algo en los años 1990, Lisp experimentó un nuevo crecimiento de interés. La mayor parte de la nueva actividad está enfocada alrededor de las implementaciones de fuente abierta del Common Lisp, e incluye el desarrollo aplicaciones y de nuevas bibliotecas portátiles. Este interés puede ser medido en parte por las ventas de la versión impresa de Practical Common Lisp (Common Lisp Práctico) de Peter Seibel, un tutorial para nuevos programadores Lisp publicado en 2004, que estuvo brevemente en Amazon.com como el segundo libro de programación más popular. El libro es accesible en línea sin costo.

Muchos nuevos programadores Lisp fueron inspirados por escritores como Paul Graham y Eric S. Raymond luchando por un lenguaje que otros consideran anticuado. Los nuevos programadores de Lisp frecuentemente describen el lenguaje como una experiencia que abre los ojos y una aclamación de ser substancialmente más productivo que otros lenguajes. Este aumento de conciencia puede ser contrastado con el «invierno de la inteligencia artificial» y el breve crecimiento del Lisp a mediados de los 1990.

En su encuesta de las implementaciones del Common Lisp, Dan Weinreb lista once implementaciones activamente mantenidas. Scieneer Common Lisp es una nueva implementación comercial que bifurcó (fork) del CMUCL con un primer lanzamiento en 2002.

La comunidad de la fuente abierta ha creado la nueva infraestructura de soporte: Cliki es un Wiki que recoge la información relacionada con el Common Lisp, el Common Lisp

directory lista recursos, el #lisp es un canal popular de IRC (con soporte por un Bot escrito en Lisp), lisppaste soporta la distribución y el intercambio y comentario de retazos de código (snippets), el Planet Lisp recoge el contenido de varios blogs relacionados con el Lisp, en el LispForum el usuario discute tópicos sobre Lisp, Lispjobs es un servicio para anunciar ofertas de trabajo y hay un nuevo servicio de noticias semanales (Weekly Lisp News).

Han sido celebrados los 50 años del Lisp (1958-2008) en LISP50@OOPSLA. Hay varias reuniones de usuario locales regulares (Boston, Vancouver, Hamburg,...), Reuniones Lisp (European Common Lisp Meeting, European Lisp Symposium) y una International Lisp Conference.

La comunidad Scheme mantiene activamente más de veinte implementaciones. Se han desarrollado en los últimos años varias significativas nuevas implementaciones (Chicken, Gauche, Ikarus, Larceny, Ypsilon). El estándar de Scheme Revised^5 Report on the Algorithmic Language Scheme fue ampliamente aceptado en la comunidad del Scheme. El proceso Scheme Requests for Implementation ha creado muchas bibliotecas y extensiones casi estándares para el Scheme. Las comunidades de usuario de implementaciones individuales del Scheme continúan creciendo. En 2003 un nuevo proceso de estandarización del lenguaje fue comenzada y condujo al estándar R^{6}RS del Scheme en 2007. El uso académico del Scheme para enseñar ciencias de la computación parece haber declinado algo. Algunas universidades ya no están usando Scheme en sus cursos preliminares de ciencias de la computación.

Hay también algunos nuevos dialectos Lisp. Notablemente: Newlisp (un lenguaje de scripting), Arc (desarrollado por Paul Graham) y recientemente Clojure (desarrollado por Rich Hickey) y NU para la programación con Cocoa de Apple.

INNOVACIONES DEL LENGUAJE

El Lisp fue el primer lenguaje de programación homoicónico: la representación primaria del código del programa es el mismo tipo de estructura de la lista que también es usada para las principales estructuras de datos. Como resultado, las funciones del Lisp pueden ser

manipuladas, alteradas o aún creadas dentro de un programa Lisp sin un extensivo análisis sintáctico(parsing) o manipulación de código de máquina binario. Esto generalmente es considerado una de las ventajas primarias del lenguaje con respecto a su poder expresivo, y hace al lenguaje favorable a la evaluación metacircular.

La ubicua estructura `IF THEN ELSE`, ahora admitida como un elemento esencial de cualquier lenguaje de programación, fue inventada por McCarthy para el uso en el Lisp, donde vio su primera apariencia en una forma más general (la estructura `cond`). Fue heredada por el ALGOL, que la popularizó.

El Lisp influenció profundamente a Alan Kay, el líder de investigación del Smalltalk, y entonces a su vez Lisp fue influenciado por Smalltalk, adoptando las características de la programación orientada a objetos (clases, instancias, etc.) a finales de los años 1970.

En gran parte debido a sus requerimientos de recursos con respecto al temprano hardware computacional (incluyendo los primeros microprocesadores), el Lisp no se hizo tan popular fuera de la comunidad de AI, como lo fueron el FORTRAN y el descendiente del lenguaje ALGOL, el lenguaje C. Lenguajes más nuevos como Java y Python han incorporado algunas versiones limitadas de algunas de las características del Lisp, pero no pueden necesariamente brindar la coherencia y la sinergia de los conceptos completos encontrados en el Lisp. Debido a su conveniencia para aplicaciones mal definidas, complejas, y dinámicas, el Lisp están disfrutando actualmente de un cierto resurgimiento del interés popular.

LISTAS

El elemento fundamental en Lisp es la lista, en el sentido más amplio del término, pues tanto los datos como los programas son listas. De ahí viene su nombre, pues Lisp es un acrónimo de «LIStProcessing».

Las listas en LISP están delimitadas por paréntesis. De aquí viene el chiste del significado de LISP: «LostInStupidParentheses» que aunque con buen humor es completamente ficticio.

Algunas de las funciones predefinidas de Lisp tienen símbolos familiares (+ para la suma, * para el producto), pero otras son más exóticas, especialmente dos que sirven precisamente para manipular listas, descomponiéndolas en sus componentes. Sus nombres («car» y «cdr») son un poco extraños, reliquias de tiempos pasados y de la estructura de los ordenadores de segunda generación, «car» devuelve la cabeza de una lista y «cdr» su cola o resto.

Lisp sigue una filosofía de tratamiento no-destructivo de los parámetros, de modo que la mayoría de las funciones devuelven una lista resultado de efectuar alguna transformación sobre la que recibieron, pero sin alterar esta última.

Uno de los motivos por los que Lisp es especialmente adecuado para la IA es el hecho de que el código y los datos tengan el mismo tratamiento (como listas); esto hace especialmente sencillo escribir programas capaces de escribir otros programas según las circunstancias.

Lisp fue uno de los primeros lenguajes de programación en incluir manejo de excepciones con las primitivas *catch* y *throw*.

Derivado del Lisp es el lenguaje de programación Logo. Sin entrar en detalles, podría decirse que Logo es Lisp sin paréntesis y con operadores aritméticos infijos.

Ejemplos

Hola mundo

```
(format t "¡Hola, mundo!")
```

Trabajando con listas

```
*************************************************************
*****
Definición de la función:

  (defun vacia (l)
    (cond ((null l) 1)    ; si la lista esta vacia devuelve 1
           (t 0)))          ; en otro caso (lista llena)
devuelve 0

Llamada a la función:

  (vacia '(1 3 4))        ; La lista no esta vacia, devolvería
0
  (vacia '())             ; La lista esta vacia, devolvería 1
*************************************************************
*****
(defun último (lista)
  (cond ((null (cdr lista)) (car lista))
        (t (último (cdr lista)))))

(último '(1 2 3 4 5 6 7))  ; devuelve el último de la lista: 7

*************************************************************
*****
;        ---FACTORIAL----
;Definición matemática
```

```
; Factorial(x) =  1                    si x=0   caso base
;                 x*factorial(x-1)     si x>0   caso recursivo

;Función factorial hecha con recursividad no final
(defun factorial (n)
   (if (= 0 n)
     1                            ; caso base
     (* n (factorial (- n 1)))))  ; caso recursivo

(factorial 4)              ;esto nos devolvería 24=4*3*2*1

*****************************************************************
*****
```