Investigación de Lenguajes - SCHEME

20 de octubre de 2013

- Adrian Aguilar
- Edinson Sanchez
- Kevin Filella

1. Introducción

El SCHEME es un lenguaje de programación que forma parte de LIPS. LISP es un conjunto o familia de lenguajes de programación que no están limitados a un único paradigma de programación, es decir son multiparadigma. Entre los lenguajes LISP más conocidos hoy en día están el Common LISP y el SCHEME (más comúnmente conocido como el infierno de los paréntesis)

A diferencia de Common LISP, SCHEME sigue una filosofía de diseño minimalista, especificando pequeñas características centrales bien definidas y juntándolas con herramientas poderosas para extender el lenguaje. Apareciendo por primera vez en 1975 en una serie de memos conocidos como los 'Lambda Papers'.

SCHEME tiene una amplia gama de aplicaciones y usos debido a su estructura compacta y su elegancia, pero su filosofía minimalista ha causado una gran discrepancia entre sus implementaciones prácticas.

SCHEME tiene un dialecto estándar oficializado por la IEEE, de los cuales el último, el R6RS fue ratificado en el 2007.

2. Características

SCHEME, debido a que pertenece a la familia LISP, es un lenguaje multiparadigma. Y aunque no tiene una sintaxis explicita para crear registros o estructuras, o para la programación orientada a objetos, si posee muchas implementaciones y herramientas que permiten estas funcionalidades.

Debido a su filosofía minimalista y a su gran variedad de herramientas e implementaciones. El "SCHEME Steering Commitee"lo ha bautizado como .[el] lenguaje mas no-portable del mundo.[a]sí como ünа familia de dialectos.[en] vez de considerarlo un único lenguaje.

Sus principales diseñadores fueron Guy L. Steele y Gerald Jay Sussman.

Su nivel de disciplina de codificación es bastante fuerte. Esto es característico de los lenguajes de programación de alto nivel viejos, ya que de estos solo FORTRAN es más viejo Entre sus implementaciones destacan el diseño e implementación de Compiladores e interpretadores. El Cálculo Lambda es una de sus características principales. Este se refiere a utilizar funciones no solo como una instancia de un procedimiento, sino también como una estructura de control y un modificador de ambiente. Esto hace que la implementación de bucles y recursividad operen de una manera bastante elegante. Hacerca de su uso actualmente, se podría decir que no es un lenguaje de uso masivo y que su niche de utilidad cada vez se hace más pequeño. Para mostrar esto veamos esta tabla que muestra los Rankings de los lenguajes de programación de acuerdo a su uso en la actualidad. Tabla tomada de Tiobe.com

Position Oct 2013	Position Oct 2012	Delta in Position	Programming Language	Ratings Oct 2013	Delta Oct 2012	Status
1	1	=	C	17.246%	-2.58%	A
2	2	=	Java	16.107%	-1.09%	A
3	3	=	Objective-C	8.992%	-0.49%	A
4	4	=	C++	8.664%	-0.60%	A
5	6	↑	PHP	6.094%	+0.43%	A
6	5	↓	C#	5.718%	-0.81%	A
7	7	=	(Visual) Basic	4.819%	-0.30%	A
8	8	=	Python	3.107%	-0.79%	A
9	23	↑↑↑↑↑↑↑↑↑↑	Transact-SQL	2.621%	+2.13%	A
10	11	↑	JavaScript	2.038%	+0.78%	A
11	18	↑↑↑↑↑↑↑	Visual Basic .NET	1.933%	+1.33%	A
12	9	↓↓↓	Perl	1.607%	-0.52%	A
13	10	↓↓↓	Ruby	1.246%	-0.56%	A
14	14	=	Pascal	0.753%	-0.09%	A
15	17	↑↑	PL/SQL	0.730%	+0.10%	A
16	13	↓↓↓	Lisp	0.725%	-0.22%	A
17	12	↓↓↓↓↓	Delphi/Object Pascal	0.701%	-0.40%	A
18	53	↑↑↑↑↑↑↑↑↑↑	Groovy	0.658%	+0.53%	B
19	19	=	MATLAB	0.614%	+0.02%	B
20	26	↑↑↑↑↑↑	COBOL	0.599%	+0.15%	B

Como podemos observar , SCHEME ni siquiera aparece en la lista, para obtener

el ranking de SCHEME tenemos que pasar a la lista extendida , he aquí a continuación, obtenida del mismo sitio especializado en ranking de lenguajes de programación, tiobe.com

Position	Programming Language	Ratings
21	R	0.553%
22	SAS	0.543%
23	Ada	0.510%
24	F#	0.499%
25	Fortran	0.474%
26	Assembly	0.471%
27	Bash	0.470%
28	Ladder Logic	0.457%
29	Logo	0.433%
30	Lua	0.413%
31	ABAP	0.394%
32	C shell	0.382%
33	Common Lisp	0.380%
34	NXT-G	0.366%
35	Scheme	0.360%
36	Scala	0.345%
37	D	0.337%
38	Prolog	0.328%
39	RPG (OS/400)	0.319%
40	PostScript	0.312%

SCHEME se encuentra en el puesto 35, aunque esto no resta a su importancia histórica ya que en su momento fue un lenguaje de suma importancia y de características únicas. Pero con el tiempo surgieron lenguajes nuevos de alto nivel que suplían y sobrepasaron las características del SCHEME

3. Historia

3.1. Carl Hewitt y el proyecto Planner

En 1971, Gerald Jay Sussman, Drew McDermott y Eugene Charniak desarrollaron un sistema llamado Micro-Planner, una parcial e insatisfactoria implementación de Planner. Sussman y Carl Hewitt trabajaron, junto con otros, en Muddle (luego MDL), una extensión de Lisp que formo un componente del proyecto Planner de Hewitt. En 1972, Sussman y McDermott desarrollaron un lenguaje basado en Lisp llamado Conniver.

En noviembre de 1972, Hewitt y sus estudiantes en el MIT desarrollaron el modelo Actor en computación, el cual fue propuesto como una solución a los problemas con Planner. Steele y Sussman luego decidieron implementar una versión del modelo Actor en MacLisp. Comenzaron a desarrollar mecanismos para crear actores y enviar mensajes.

3.2. Implementacion del modelo Actor y el nacimiento de Scheme

Sussman y Guy L. Steele luego implementaron el modelo Actor en el cálculo lambda, llamando a su sistema de modelación: Schemer. Este nombre fue luego reducido a 'Scheme' ya que hicieron uso del sistema operativo ITS, que limitaba los nombres de fichero a seis caracteres. Luego, ambos concluyeron que los Actores y los Closures eran, para propósitos de sus investigaciones, un concepto idéntico y procedieron a eliminar el código redundante. Luego de la eliminación, se dieron cuenta que habían creado un pequeño, y muy capaz, dialecto de Lisp.

Hewitt estuvo escéptico de las capacidades del cálculo lambda como base de la programación y dijo, "La actual situación es que el cálculo lambda es capaz de expresar ciertos tipos de estructuras de control paralelo y secuencial, pero, en general, no es capaz de expresar la concurrencia que expresa el modelo Actor. Por otro lado, el modelo Actor es capaz de expresar todo lo del cálculo lambda, y mas."Hewitt también ha sido fuerte en las criticas sobre Scheme en base a su falta de manejo de excepciones y su dependencia a lo que se llaman continuation functions.

3.3. Publicación y Estandarización

Scheme es un lenguaje de programación que enfatiza la aplicación de funciones, o lenguaje funcional, que tiene sus raíces en el lenguaje de programación Lisp, siendo un dialecto de este. Scheme tuvo sus inicios en el periodo de 1975 hasta 1980, cuando Guy L. Steele y Gerald Jay Sussman, del MIT AI Lab, publicaron nueve artículos llamados Lambda Papers durante este tiempo.

Aunque una de las características mas importantes de destacar sobre Scheme es el minimalismo, Sussman y Steele expresaron que esta no fue una meta intencional durante su creación. "Nosotros en realidad estábamos tratando de construir algo complicado y rebuscado, favorablemente, habíamos creado algo que, además de cumplir con todas nuestras metas, era mucho más simple de lo intencionado... Nos dimos cuenta que el cálculo lambda puede servir como la base de un poderoso lenguaje de programación."(Gerald Jay Sussman y Guy L. Steele, Jr. (Diciembre 1998). "The First Report on Scheme Revisited").

Scheme tiene un estándar oficial de la IEEE y un estándar de facto llamado Revised n-th Report on the Algorithmic Language Scheme (comunmente abreviado RnRS). El estándar mas implementado hasta la fecha es el R5RS de 1998, y el nuevo estándar, R6RS, fue ratificado el 2007.

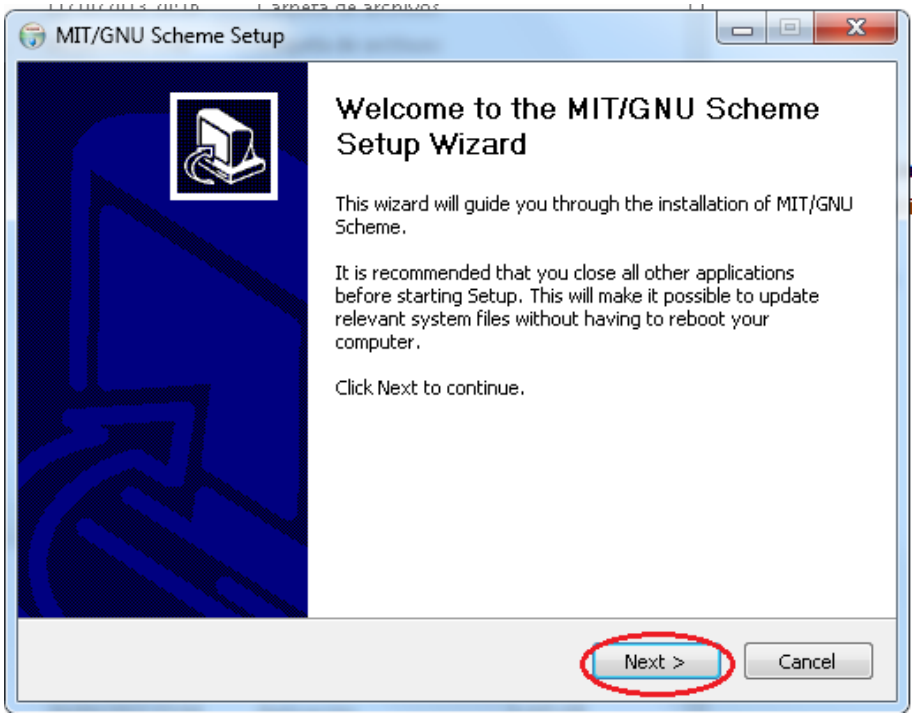
4. Tutorial de Instalación

4.1. Debemos ir al siguiente link (http://www.gnu.org/software/mit-scheme/) y descargar el paquete de instalación del lenguaje SCHEME correspondiente al sistema operativo en la q se va a trabajar, en este caso yo instalare el paquete de Windows 7. lo elegimos e inmediatamente comenzara la descarga del paquete de instalación

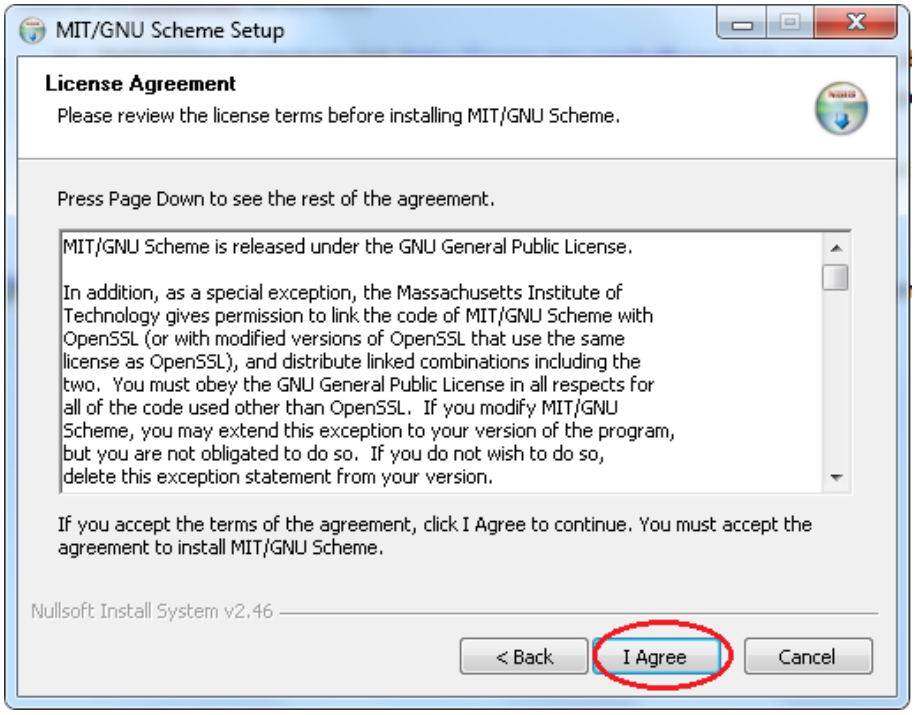
Stable release 9.1.1

File	Arch	Instructions	Notes
Unix binary	i386	unix installation	
Unix binary	x86-64	unix installation	
OS X binary	i386		Compiled on OS X 10.6.
OS X binary	x86-64		Compiled on OS X 10.6.
Windows binary	i386	Windows installation	Compiled on Windows 7 using Open Watcom C/C++ 1.9.
Portable C	(any)	how to build	For use on any unix system.
Source (.tar.gz)			For unix systems; uses linefeeds as line delimiters.
Change log			
MD5 checksums			

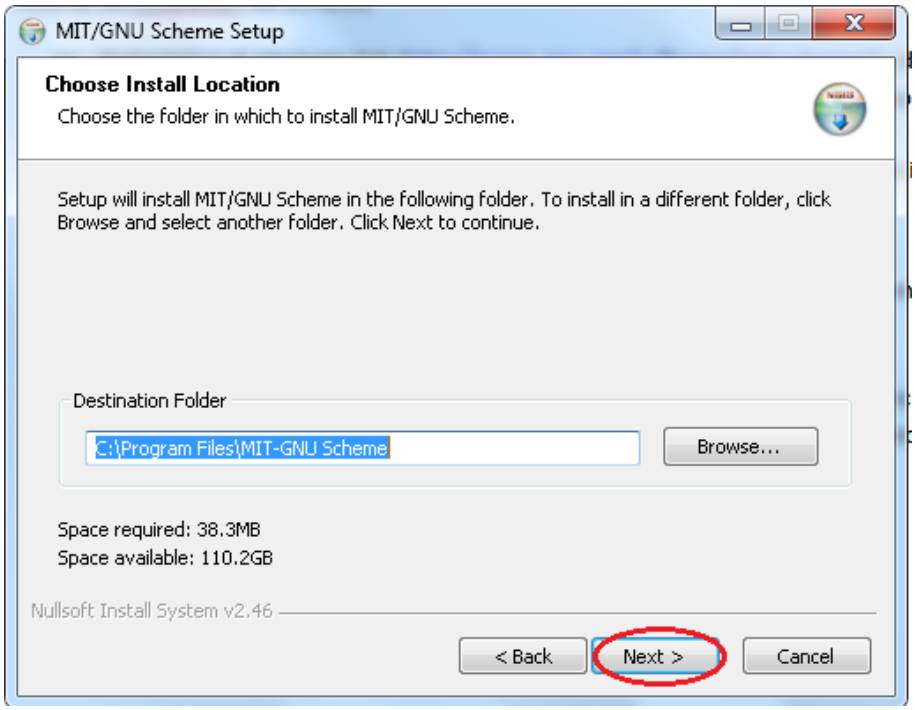
4.2. Ejecutamos el paquete de instalación de SCHEME y aparecerá la siguiente ventana donde le damos en NEXT



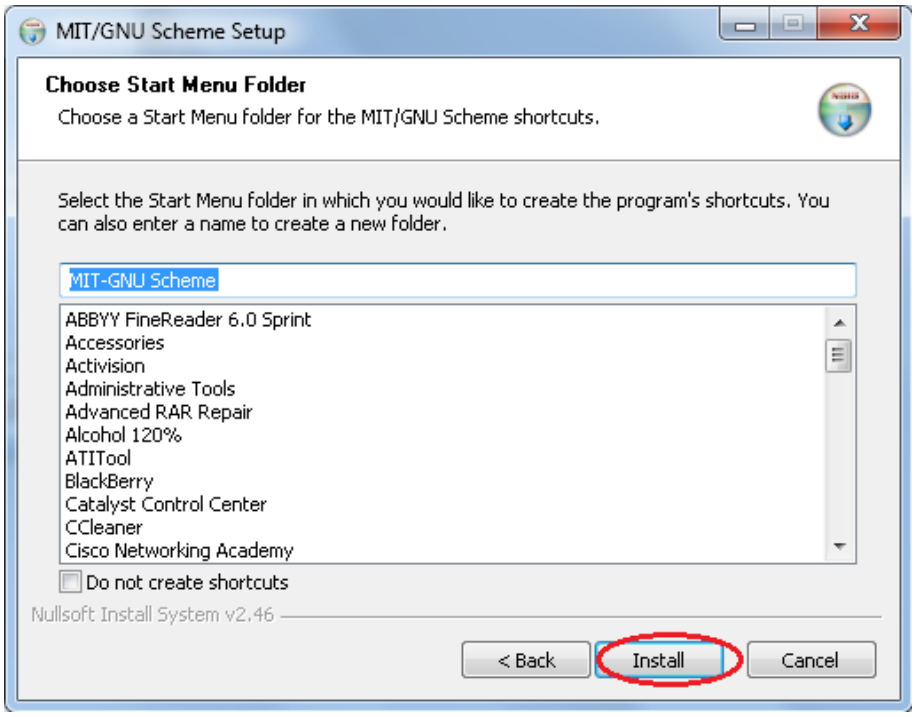
4.3. Ahora nos aparecerá una nueva ventana donde estarán los términos de licencia del programa, si está de acuerdo le da clic en I AGREE, caso contrario atrás o cancelar



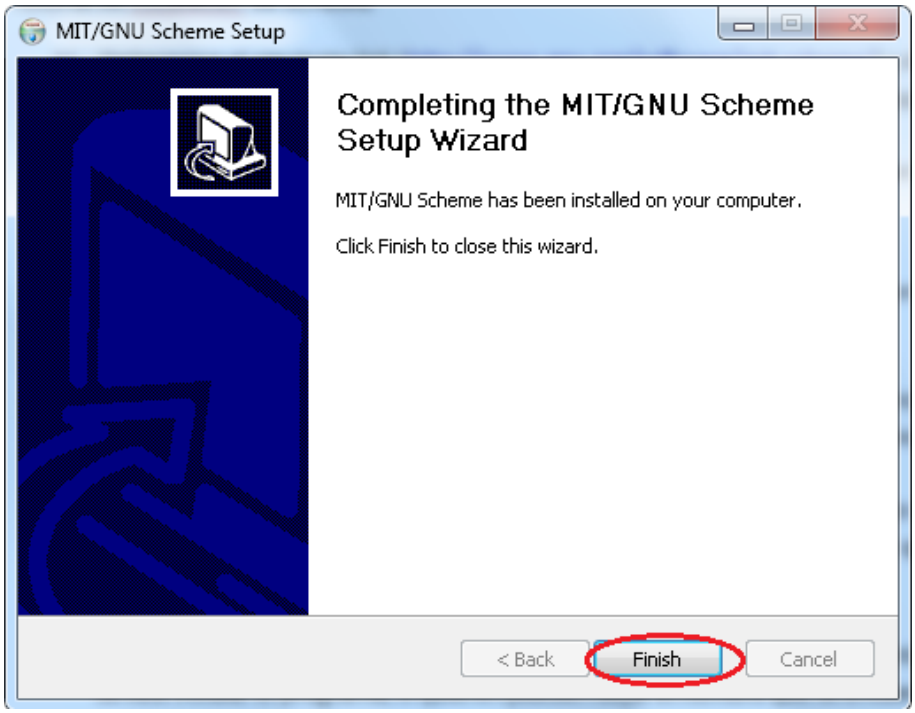
4.4. Nos aparecerá otra ventana donde debemos ingresar una dirección e el disco duro donde queremos q el programa se instale, si quiere puede dejar la dirección que se encuentra escrita por defecto y le da clic en NEXT



4.5. Aparecerá una nueva ventana donde podemos cambiar el nombre a la carpeta donde se va a instalar el programa, si quieren pueden dejar el nombre que se encuentra por defecto y dar clic en INSTALL

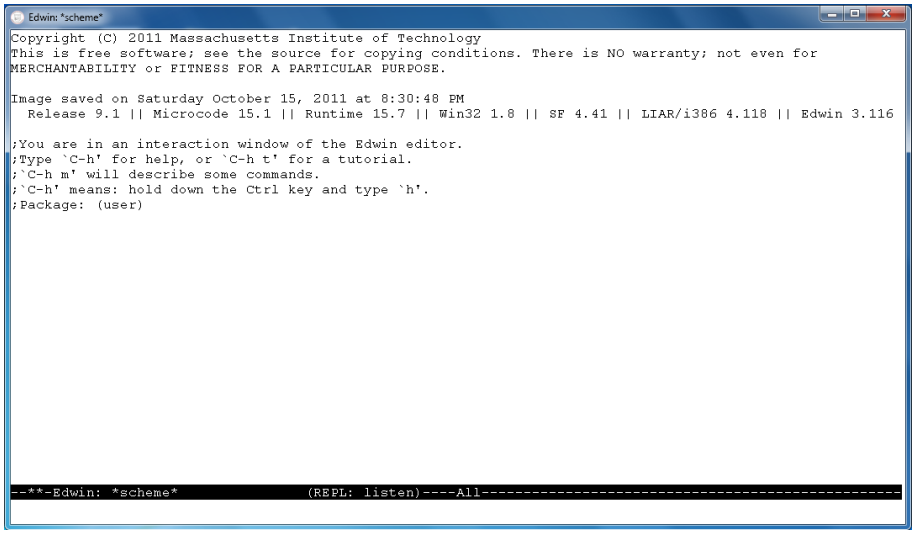


4.6. Comenzara a instalarse el programa hasta que la barra llegue al 100 y aparecerá una nueva ventana donde le damos clic en FINISH para confirmar que el programa se ha instalado correctamente



4.7. Ahora vemos en nuestro escritorio y nos va a aparecer un nuevo icono que le pertenece al programa SCHEME recién instalado, le damos doble clic y se abrirá

4.8. Finalmente tenemos instalado en nuestro ordenador el lenguaje SCHEME listo para trabajar



5. Hola Mundo y otros Programas Introductorios

```
> (display "Hola mundo")
Hola mundo
```

'display' se utiliza para mostrar texto en pantalla. Este código muestra 'Hola mundo' en pantalla.

```
> (sin 90)
1
```

Este código muestra el seno de 90 grados en pantalla: 1.

```
> (map + '(5 2 1) '(3 4 4))
(8 6 5)
```

'map' recibe un procedimiento y lo utiliza en los elementos de una lista, retornando una lista de resultados. En este caso, utiliza el operador suma y retorna la suma de las dos listas: (8 6 5).

```
> (define (cuadrado x) (* x x))
> (cuadrado 10)
100
```

En este código, hemos declarado una función llamada 'cuadrado' que recibe un parámetro 'x'. El cuerpo de tal función esta después de su declaración, y consiste en elevar 'x' al cuadrado usando el operador de multiplicación. En la siguiente línea, llamamos a la función 'cuadrado' y le enviamos '10' como parámetro, retornando 100 como resultado.