Universidad Nacional Autónoma de México

Facultad de Ciencias

Práctica 4: Teoría

Ximena Lezama Hernández
Ángel Gladín García

¡Nos veremos pronto Haskell!

29 de marzo de 2017

1. Preguntas

1. Si al terminar la evaluación de una expresión no resulta un número entero. ¿Es una evaluación válida o la máquina lanza mensaje de error? Justifica la respuesta con tus propias palabras.

   Lanza un error, ya que en sust, al sustituir las variables de la expresión aritmética, si la variable no es sustituida por un entero, lanza el error "No es posible hacer sustitución."

2. Investiga tres lenguajes de programación que al menos soporten los paradigmas imperativo y funcional e indica si su mecanismo de evaluación es perezosa o no.

   a) SCALA: Es multi-paradigma soporta los paradigmas imperativo, funcional, orientado a objetos y su mecanismo de evaluación es perezosa.

   b) Ocaml: Es multi-paradigma soporta los paradigmas imperativo, funcional y su mecanismo de evaluación es perezosa.

   c) F Sharp: Es multi-paradigma soporta los paradigmas imperativo, funcional, orientado a objetos y no es de evaluación perezosa por default como Haskell, pero de manera explicita puedes usar evaluación perezosa.

3. Plankalkül es considerado el primer lenguaje de programación de alto nivel, menciona al menos 5 características que tu consideres interesantes teniendo en cuenta que fue el primero.

   1. Que los datos se escriben en punto flotante, punto fijo, complejos.
   2. Hay bucles.
   3. Que Maneja arrays, registros, estructuras de datos jerárquicas y listas de parejas.
   4. Poderosas operaciones con listas y parejas de listas.
   5. Es un lenguaje de programación imperativo de alto nivel.

4. ¿Cuál fue el primer lenguaje de programación funcional de la historia?

   Fue Lips el primer lenguaje de pregramación funcional en 1958, desarrollado por Steve Russell, Timothy P. Hart, y Mike Levin. Lisp fue

creado originalmente como una notación matemática práctica para los programas de computadora, basada en el cálculo lambda de Alonzo Church.

5. ¿Qué es la sintaxis abstracta y la sintaxis concreta en el estudio de lenguajes de programación?

   La sintaxis concreta de un lenguaje de programación está definida por una gramática libre de contexto. Consiste en un conjunto de reglas (producciones) que definen el aspecto de los programas para el programador.
   Mientras que la sintaxis abstracta de una implementación es el conjunto de árboles utilizado para representar programas en la implementación. Los arboles de sintaxis abstracta, o simplemente arboles sintácticos difieren de los arboles de análisis sintáctico en que las distinciones superficiales de forma, sin importancia en la traducción, no aparecen en los arboles sintácticos. La sintaxis abstracta define el aspecto de los programas para el evaluador / compilador.