Lenguajes de Programación Guava - Version 1

Definición del Lenguaje

Guava es un lenguaje imperativo fuertemente tipado, con sintáxis similar a la del lenguaje C o Java. Además de ofrecer tipos de datos primitivos: enteros, flotantes, booleanos, caracteres y cadenas de caracteres, Guava proporciona tipos compuestos de datos similares a los de C: estructuras y uniones. El lenguaje también ofrece funciones implementadas y definidas por el programador.

Como se explico anteriormente Guava tiene similitudes con lenguajes como C/C++, Java, Go o Rust por lo que si se viene con un trasfondo de estos lenguajes se estará cómodo, sin embargo existen diferencias importantes como el modo de asignación, los arreglos dentro del lenguaje (no apuntadores), operadores especiales para facilidad de programacion y palabras reservadas expresivas.

[Colocar aquí los puntos característicos de Guava (plus con los arreglos)].

Gramática

[Especificar la gramática del lenguaje en esta sección]

PROGRAM -> BLOQUEPRINCIPAL

BLOQUEPRINCIPAL -> BLOQUEDECLARE LFUNCIONES

BLOQUEDECLARE -> declare { LVARIABLES } |

LVARIABLES -> TIPO LVAR; LVARIABLES | TIPO LVAR;

LVAR -> VAR | VAR, LVAR

LFUNCIONES -> FUNCION LFUNCIONES | FUNCION

FUNCION -> function TIPO ID (LPARAM){ BLOQUEDECLARE LISTAINSTRUCCIONES return VAR; } | function void ID (LPARAM){ BLOQUEDECLARE LISTAINSTRUCCIONES }

LPARAM -> | TIPO VAR , LPARAM | TIPO VAR

LISTAINSTRUCCIONES -> INSTRUCCION ; LISTAINSTRUCCIONES |

INSTRUCCION -> ASIGNACION | LOOPFOR | LOOPWHILE | LLAMADAFUNCION | SELECTORIF

ASIGNACION -> VAR := EXP

LOOPFOR -> for (VAR ; EXP ; ASIGNACION){ BLOQUEDECLARE LISTAINSTRUCCIONES } | for (VAR ; EXP ; EXP){ BLOQUEDECLARE LISTAINSTRUCCIONES }

LOOPWHILE

EXP -> EXPBIN | EXPUN | VALOR | VAR | (EXP)

EXPBIN -> EXP and EXP // Expresiones booleanas binarias. | EXP or EXP | EXP <=> EXP | EXP > EXP | EXP < EXP | EXP >= EXP | EXP <= EXP | EXP = EXP | EXP != EXP | EXP + EXP // Expresiones numericas binarias | EXP - EXP | EXP * EXP | EXP / EXP | EXP div EXP | EXP mod EXP | EXP ** EXP

```
EXPUN -> not EXP
      |  - EXP
      |  EXP ++
      |  EXP --

VALOR -> true
      |  false
      |  STRING
      |  CHAR
      |  INT
      |  REAL
```

Estructura de un programa

Un programa en Guava se estructura en un conjunto de definiciones de funciones, procedimientos, estructuras, declaraciones de variables (en este caso, consideradas globales) y un procedimiento principal *main. [Se debe especificar que no existe orden en los 'bloques de definicion' de la estructura del lenguaje]*

Cada bloque de código dentro de un programa estará constituido a su vez por una serie de instrucciones separadas por el secuenciador **;** menos la última instrucción del bloque.

Elementos del lenguaje

Identificadores

Los identificadores de variables son cadenas de caracteres que incluyen las letras que van desde la **a** a la **z** (mayúsculas y minúsculas incluidas) sin incluir la letra ñ, los números del **0** al **9** y el símbolo __. Los identificadores no pueden iniciar con un número y son sensibles a mayúsculas, teniendo entonces que **gua** es un identificador diferente a **Gua** y éste a su vez a **gUA**.

Variables

Una variable es un valor que puede ser alterado durante la ejecución de un programa en Guava. Las variables son nombradas a partir de un identificador y se clasifican en:

- **Variables definidas por valor**: Son variables cuyo valor es guardado de forma estática en la memoria reservada para la ejecución del programa, una vez instanciada sólo podrá ser cambiado su valor a través de ella misma. Para declarar una variable definida por valor se especifica:

```
<tipo_dato> <identificador>
```

- **Variables definidas por referencia**: Son variables cuyo valor es guardado en una dirección de memoria que puede ser accedida por más de una instancia de variable, es decir, pueden existir varias instancias que apunten a una misma dirección de memoria y por ende, posean el mismo valor. Modificar el valor de una variable definida por referencia se traduce en modificar el valor de todas las instancias que apunten también a su dirección de memoria. Para declarar una variable de este tipo se especifica:

```
<tipo_dato> var <identificador>
```

Siendo `var` una palabra reservada del lenguaje.

Tipos de datos

Guava dispone de los siguientes tipos de datos primitivos:

- `integer` : Números enteros con signo representables en 32 bits.
- `real` : Números en punto flotante con signo representables en 32 bits.
- `boolean` : Booleanos con los valores constantes `true` y `false`.
- `character`: Representa un caracter único. Incluye las letras de la **a** a la **z**, los números del **0** al **9** y caracteres especiales.
- `string` : Representa una cadenas de caracteres. Incluye una combinación de las letras de la **a** a la **z**, los números del **0** al **9** y caracteres especiales.

Además el lenguaje ofrece la creación de los tipos compuestos:

- `record` : Define un conjunto de variables agrupadas por un nombre en un bloque de memoria. Las variables que conforman la estructura pueden ser accedidas por un apuntador.
- `union` : Define un conjunto de variables agrupadas por un nombre en un bloque de memoria, pero a diferencia de `record` cada una de las variables del bloque comparten la misma dirección de memoria. Las variables de la unión pueden ser accedidas por un apuntador.

Los nombres de cada tipo de datos son palabras reservadas del lenguaje, por ende, no pueden ser utilizadas como identificadores de variables.

Arreglos

El lenguaje dispone de estructuras de datos de tipo arreglo. Los arreglos son homogéneos y ordenados linealmente por un número entero o índice.

La siguiente parte debe ser revisada y consultada:

Los arreglos en Guava son manejados estáticamente, al declarar una estructura de datos de este tipo se debe especificar el tamaño de la siguiente manera:

```
<tipo_dato> array[N] <identificador>
```

Siendo `N` el tamaño del arreglo.

fin de revisión y consulta

Operaciones asociadas a los Arreglos: ***[sujeto a cambios]*** Aquí entonces se debe de especificar todas las cosas finas que podemos hacer con los arreglos en Guava para que se diferencie de los demás lenguajes de programación, :). *[o también redirigir a la sección de "operaciones con arreglos"]*

Funciones

Las funciones en Guava son declaradas y definidas en forma de bloques de código. Poseen una firma que indica el nombre de la función acompañado de un tipo de retorno (o **void** en caso de no retornar ningún valor) y una lista de parámetros formales acompañados de su tipo de dato respectivo de la siguiente manera:

```
<tipo_dato_retorno> function <identificador>(<tipo1> <paramétro1>,
<tipo2> <parámetro2>, ..., <tipoN> <parámetroN>)
```

Consideraciones generales sobre las funciones:

- Las funciones son definidas en el espacio global del programa. No es posible definir funciones dentro de un bloque de código de otra función.
- La lista de parámetros formales puede tener de 0 a N elementos, siendo posible definir funciones que no reciban parámetros formales para su aplicación.
- Los tipos de datos de los parámetros formales y de retorno de una función pueden ser tipos primitivos, compuestos y arreglos definidos en el lenguaje.
- Los identificadores empleados en la lista de parámetros formales deben de ser diferentes unos de otros, al igual que diferentes a los identificadores de las variables locales a la función.
- Las funciones en Guava pueden ser recursivas.

Expresiones

En Guava, una expresión es cualquier enunciado que pueda ser evaluado y devuelve un valor.

Operaciones con Enteros: Guava soporta las siguientes operaciones aritméticas sobre el conjunto de los números enteros, con el siguiente orden de precedencia (de menor a mayor y operandos en una misma línea poseen igual precedencia):

- `--, ++ :: integer -> integer` (decremento e incremento prefijo)
- `+, - :: integer -> integer -> integer`
- `*, div, mod :: integer -> integer -> integer`
- `** :: integer -> integer -> integer`
- `- :: integer -> integer` (menos unario)
- `--, ++ :: integer -> integer` (decremento e incremento sufijo)

Los operadores de suma, resta, multiplicación, cociente, módulo, decremento, e incremento sufijo asocian hacia la izquierda, mientras que los operadores de potenciación, decremento e incremento prefijo asocian hacia la derecha.

La comparación de enteros se realizará con los siguientes operadores:

- `>=, <=, >, <, =, != :: integer -> integer -> boolean`

Operaciones con Reales: Las operaciones correspondientes a la aritmética de reales son las siguientes (con orden de precedencia de menor a mayor):

- `+, - :: real -> real -> real`
- `*, \ :: real -> real -> real`
- `- :: real -> real` (menos unario)

Los operadores de suma, resta, multiplicación y división asocian hacia la izquierda.

La comparación de reales se realizará con los operadores:

- `>=, <=, >, <, =, != :: real -> real -> boolean`

Operaciones con Caracteres: Guava proporciona las siguientes operaciones sobre caracteres:

- `toInt ::`