Especificación del lenguaje BLAH (Best Language At Home)

Introducción

Este documento explica la expecificación del lenguaje BLAH. Es un lenguaje simple, de alcance estático, imperativo, sin orientación a objetos con un sistema de tipos estático y similar a C/C++.

Léxico

El léxico del lenguaje está basado en el lenguaje natural. Las palabras reservadas del lenguaje, que no pueden ser usadas por el programador ni redefinidas son:

```
Integer, Boolean, Floating, Char, String, for, from, to,
while, do, if, then, else, return, stop, next, print, read,
register, union, Arrayof, true, false, plus, minus,
times, dividedby, void, and, or, is, not.
```

Un identificador es una secuencia de letras, dígitos y underscores, comenzando por una letra. BLAH es sensible a la capitalización, por ejemplo, foo y fOO son identificadores diferentes. Todos los tipos deben comenzar por una mayuscula, mientras que las variables y las funciones con una minuscula.

Los espacios en blanco (i.e espacios, tabs, etc) sirven para separar tokens pero de resto son ignorados. Palabras reservadas e identificadores deben estar separados por un espacio en blanco, o un token que no sea palabra reservada ni otro identificador. En otras palabras, `arrayofinteger` es un identificador y `array)of)integer` se reconoce como 5 tokens.

Un número entero puede ser especificado en decimal (base 10), octal (base 8), o hexadecimal (base 16). Un entero decimal es una secuencia de dígitos decimales. Un entero en octal se especifica con un `0` (zero) al principio del número y es seguido por una secuencia de dígitos octales. Por ejemplo `01`, `012`, `05369`. Un entero hexadecimal debe empezar con `0x` o con `0X` y seguido de una secuencia de dígitos hexadecimales. Los dígitos hexadecimales incluyen los dígitos decimales y las letras del abecedario de la `a` a la `f`, en mayúsculas o minúsculas por igual. Para dejar constancia, los siguientes son enteros válidos:

```
8, 16, 01234, 0x5ad89, 0X4Eb23, 02342
```

Un número flotante es una secuencia de dígitos decimales, un punto, y continúa con otra secuencia de dígitos. Por tanto, ni `.12` ni `12.` son válidos, pero 12.0 y 0.12 si lo son. Un número decimal puede tener un exponente opcional, que

se denota con una `e` (minúscula o mayúscula) al final del número seguido del exponente, por ejemplo, `55.7E+2`, `0.34e-8`. Como se explicó más arriba `34.E+2` y `.12e-15` son inválidos. Si no se especifica el signo del exponente el positivo (+) es asumido.

Una secuencia de caracteres encerrada entre comillas simples constituyen un string literal, es inválido escribir comilla simple en un string de este tipo.
Una secuencia de caracteres entre comillas dobles requieren escapar signos especiales para tomarlos como parte de la cadena. Un string debe empezar y terminar en la misma línea no puede ser escrito en varias líneas. Para ilustrar lo anterior:

```
'Aunque este string está escrito para que se vea deliberadamente
largo y ocupe más de una línea, debe empezar y terminar en la
misma línea para que pueda ser válido en el lenguaje BLAH'

'Este un string literal donde \n no es un salto de línea'

"En los strings con doble comilla \n es un salto de línea"

"Este es un string que requiere de escapar la doble comilla (\")
para que pueda ser considerada parte de la cadena"

'Este string es inválido pues tiene una comilla simple "'"
extra en el medio'
```

Signos de puntuación usados por el lenguaje incluyen:

```
+ - * / % < <= > >= = == != && || ! , . [ ] ( ) { } " ' /#
```

Un comentario de una línea comienza con `//` y se extiende hasta el final de la línea. Como en C/C++, un comentario de bloque empieza por `/*` y termina con `*/`. Cualquier símbolo es permitido en el comentario excepto `*/` que termina el comentario actual. Así como en C/C++ los comentarios no se anidan.

Estructura de los programas

Un programa de BLAH es una secuencia de declaraciones, donde cada declaración establece una variable o una función. Una declaración se refiere a una sentencia que establece la identidad de dichas variables o funciones, mientras que una definición se refiere a la descripción total de su funcionamiento o cuerpo. Para nuestros propósitos, las declaraciones y definiciones pueden estar juntas o separadas. En caso de que sean separadas, las declaraciones deben estar antes o en la misma línea de las definiciones o inicializaciones. El programa debe tener una función principal llamada `main` que no toma argumentos y retorna un tipo `integer`. Esta función sirve de punto de entrada para la ejecución del programa.

Alcance

BLAH soporta varios niveles de alcance. El alcance a nivel principal es el alcance global. Un nuevo bloque de alcance anidado se define con un par de llaves `{ }`. Cada función define un alcance local. Los alcances internos tapan los alcances más externos. Por ejemplo, una variable definida en una función tapa una variable con el mismo identificador en el alcance global.

Las siguientes reglas se aplican para los identificadores:

- Al entrar a un nuevo alcance, todas las declaraciones en ese alcance son inmediatamente visibles.
- Los identificadores dentro de un nivel deben ser únicos (i.e. no pueden haber dos funciones o variables del mismo nombre).
- Los identificadores redeclarados en un alcance anidado tapan las declaraciones del nivel anterior.
- Declaraciones en el nivel global pueden ser accesadas en cualquier nivel al menos que sean tapadas por otro identificador.
- Todos los identificadores deben ser declarados.
- Para poder usar un identificador este debe ser declarado antes. No se permite definir funciones o acceder variables antes de ser declaradas.

Tipos

Los tipos base en BLAH son `integer`, `character`, `floating`, `char`, `string` y `void`. BLAH soporta tipos nuevos para el caso de registros y uniones (la especificación está más adelante). Se pueden construir arreglos de cualquier tipo que no sea `void`. El tipo `string` es manejado como un arreglo de `char`.

Variables

Las variables pueden ser declaradas de cualquier tipo base, arreglo, registro o unión. Variables declaradas fuera de las funciones son de alcance global y pueden ser inicializadas pero no pueden efectuarse asignaciones a variables en este alcance. Variables declaradas como parámetros formales o en el cuerpo de una función tienen alcance local.

Arreglos

Los arreglos de BLAH son colecciones homogéneas, indexadas linealmente. Si se declara un arreglo de N elementos el primer elemento es el 0 y el N-esimo, puede ser accedido con el índice N–1 o con el índice –1. En otras palabras, es posible acceder a los elementos de un arreglo desde el último hasta el primero con números negativos, siendo el último el indexado con –1 y el primero el indexado con -N. Para ilustrar:

```
arrayof [5] char ejemplo0 is [#a,#b,#c,#d,#e].
```

El primer elemento `a` puede ser accedido como `ejemplo[0]` o como `ejemplo[-5]` y el elemento `e` puede ser accedido como `ejemplo[4]` o como `ejemplo[-1]`.

Si el tamaño dispuesto para el arreglo es diferente a la cantidad de elementos listados se generará un warning y se tomará la menor entre estas dos cantidades para inicializar esa cantidad de elementos. Por ejemplo:

```
arrayof [10] char ejemplo1 is [#a,#b,#c].
```

En ese caso se inicializará un arreglo de tamaño 10, cuyos 3 primeros elementos son `a`, `b` y `c`; el resto son null.

```
arrayof [3] char ejemplo2 is [#u,#v,#w,#x,#y,#z].
```

En ese caso se inicializará un arreglo de tamaño 3, cuyos 3 primeros elementos son `u`,`v` y ,`w`.

En BLAH es posible declarar arreglos de varias dimensiones. Para ello las dimensiones se escriben en una arreglo de enteros de la siguiente manera:

```
arrayof [3,2] integer ejemplo3 is [[1,2],[3,4],[5,6]].
arrayof [2,3] integer ejemplo3 is [[1,2,3],[4,5,6]].
```

Cadenas de Caracteres

Los programas incluyen cadenas de caracteres constantes, es posible comparar strings, imprimirlos, etc. Se planea añadir funciones que faciliten la manipulación de este tipo de datos. Las cadenas de caracteres están implementadas como arreglos de caracteres al estilo C.

- La asignación de cadenas de caracteres es profunda.
- Las cadenas de caracteres pueden ser pasadas como parámetros y retornadas de las funciones.
- La comparación de strings (`==` y `!=`) es profunda y sensitiva a mayúsculas.
- Los strings son un alias para arreglos de caracteres.

Funciones

La declaración de una función incluye el nombre de la función, su tipo de retorno asociado asi como los parámetros formales.

- Las funciones son globales, no se permite la declaración anidada de funciones.
- Las funciones pueden tener cero o más parámetros formales.
- Los parámetros formales pueden ser de cualquier tipo base (excepto `void`), arreglos o definidos como registros o uniones.
- Los parámetros formales deben tener nombres distintos.
- Los parámetros formales tienen alcance local en la función, si otro bloque dentro de ésta es definido, variables dentro de ese bloque pueden tapar a los parámetros formales.
- Una función puede volver cualquier tipo base, arreglo, registro o unión. `void` es usado para indicar que la función no retorna ningún valor.
- Si una función retorna un tipo distinto a `void`, el valor retornado debe ser compatible con ese tipo.
- Si una función retorna `void` se puede usar la instrucción return sola.
- Funciones recursivas son permitidas.

Invocacion de funciones

Los parámetros de BLAH son pasados por valor en el caso de los tipos básicos y por referencia en el caso de registros y uniones. El valor de retorno se pasa por valor y en el caso de tipos compuestos esos "valores" son referencias.

- El número de argumentos pasados en la llamada de una función deben ser iguales a la cantidad de parámetros formales y estar en el mismo orden.
- Se permiten parámetros con valores por defecto.
- El tipo de cada argumento debe ser igual al tipo de los parámetros formales.
- Los argumentos son evaluados de derecha a izquierda.
- Una función retorna el control de la ejecución al llamador cuando se ejecuta la instrucción return.
- La llamada de una función se evalúa como el tipo de dato que devuelve.

Registros y Uniones

Los registros son un tipo compuesto y en BLAH deben ser declarados como tipo antes de poder declarar variables de ese tipo. La declaración de un registro o unión se hace de manera similar a C/C++. Ejemplos:

```
union Un {
    integer c,
    float d,
    char e
}

register ASDF{
    integer a,
    float c
}
```

Equivalencia de tipos y compatibilidad

BLAH es un lenguaje fuertemente tipeado: un tipo específico es asociado a cada variable y la variable solo puede contener valores que se encuentren en el dominio de ese tipo. Si el tipo A es equivalente al B, ambas expresiones pueden ser sustituidas en cualquier situación. Dos tipos bases son equivalente si y solo si son exactamente del mismo tipo. Dos arreglos son equivalentes si y solo si son del mismo tipo base.

La equivalencia de tipos en BLAH es por nombre, es decir dos variables son equivalentes si sus tipos poseen el mismo nombre.

Asignación

La asignación BLAH es una instrucción mas no es una expresión, en otras palabras, no retorna un valor. La asignación se hace mediante la palabra reservada `is` o mediante el símbolo tradicional `=`.

Efectuar asignaciones en alcance global a variables declaradas o inicializadas en este mismo alcance no está permitido.

Estructuras de control

BLAH soporta estucturas de control similares a otros lenguajes. El tradicional `if then else if then else` como estructura de selección y en los ciclos tenemos:

* La estructura ‘for‘ en sus tres tipos: iteración por variable, por arreglo y por rango.
* La estructura ‘while do‘ y ‘do while‘.

Otras consideraciones tomadas en estas estructuras son las siguientes:

* Las condiciones de estas estructuras deben ser de tipo ‘boolean‘. No se permiten usar enteros.
* La instruccion ‘break‘ solo puede aparecer dentro de los ciclos.
* Se provee de un campo step para los ‘for‘ con rango para definir el paso por vuelta.

Expresiones

* BLAH no soporta cohersión de tipos entre enteros y flotantes.
* Entre tipos diferentes solo se permite operar con un entero y un double, el resultado sera un double.
* El operando para el menos unario debe ser ‘integer‘, ‘double‘ o ‘float‘. El resultado sera del mismo tipo que los operandos.
* Los operandos para operadores relacionales (<,>,<=,>=) deben ser ‘integer‘, ‘double‘ o ‘float‘. El resultado es ‘boolean‘.
* Para el caso de los operadores de equidad (=,!=) deben ser de tipo equivalente. El resultado sera de tipo ‘boolean‘.
* Los operandos para los operadores binarios y unarios logicos deben ser ‘boolean‘ (&&,|| y !). El resultado sera ‘boolean‘.
* Los operandos son evaluados de izquierda a derecha excepto los relacionales que son no asociativos.

La precedencia de los operadores es, de mayor a menor:

```
->                  Acceso a estructuras
- !                 Operadores unarios
* /                 Multiplicacion y division
+ - and or          Suma Resta And y Or
== != < > <= >=     Operadores Relacionales
```

Lo que no tiene el lenguaje hasta esta última versión

Algunos de estos items estan en la especificación pero se harán en el siguiente curso, sin embargo, algunos de estos items si son solicitados para la presente entrega pero no se pudieron terminar.

* No se pueden declarar firmas de funciones sin el cuerpo.
* El arbol que se genera despues del parseo no esta optimizado

* No se verfiica que la variable de control del ‘for‘ no sea usada como ‘lvalue‘
* No se verifica que la función tenga un return para cada rama de ejecución posible
* No se chequea el uso de variables no inicializadas.
* No se detecta codigo no alcanzable.
* Los pasajes por valor y referencia no se encuentran implementados.
* Los enteros con formato hexadecimal y octal no se agregaron al lexer.
* No se chequea que exista una función ‘main‘.