Sintaxis del lenguaje KASKELL

ÁLVARO GARCÍA TENORIO

MIGUEL PASCUAL DOMÍNGUEZ

- Identificadores y ámbitos de definición
  - Declaración de variables simples

    tipo identificador
  - Arrays de cualquier tipo

    [tamaño] tipo identificador

    de esta forma la sintaxis de arrays de arrays y array de varias dimensiones es indistinguible

    ARRAYS DE ARRAYS ~> [tamaño] ([tamaño] tipo) identificador ≃ [tamaño] [tamaño] tipo ident.

    ([tamaño] tipo = tipo; [tamaño] ([tamaño] tipo) = tipo)

    los paréntesis y espacios deberán ser opcionales.

    (Opcionalmente podríamos permitir tamaños indefinidos [ ] tipo identificador)
  - Bloques anidados.

    Los bloques irán entre llaves { código } (Bloque) y serán anidables.
  - Punteros

    K-tipo identificador (puntero de tipo tipo)
  - Registros ~> Hemos asumido que se refiere al equivalente a los structs en C

    struKt identificador {

    declaración de variables

    };
  - Procedimientos y funciones

    identificador : tipo1 X ... X tipon → tipoRetorno (X: equis mayúscula)

    (en otra línea) ~> (id1, ..., idn) |-> { ... expresión de tipo tipoRetorno } (return)

    ¿Dar soporte a recursión? ~> Llamada a función: identificador (id1, ..., idn);
  - Un módulo es un archivo de código sin más.
  - Cláusula de importación de módulos

    inKlude nombreArchivo

- Tipos
  - enteros ~> int, posibles valores números enteros positivos o negativos.
  - booleanos ~> bool, posibles valores true y false
  - Operadores infijos

    suma (prior 1) +, resta (prior 1) −, producto (prior 2) *, división (prior 2, asociativo por la izquierda) /,

    módulo (prior 2, asociativo por la izquierda) mod, potencia (prior 3, asociativo por la izquierda) ^, or (prior 1) or, and (prior 2) and
  - Tipos sin nombre

    KK identificador = { declaración de variables asignadas };

- Instrucciones ejecutables
  - Asignación de variables
    identificador = expresión ;
    identificador [posicion] = expresión ;
  - Condicional (como en C)

```
if (condicion) {
} else {
}
```

```
if (condicion) {
}
```

```
if (condicion) {
} else if (condicion) {
} else if (condicion) {
   .
   .
   .
} else {
}
```

  - Bucles

```
while (condicion) {
   ...
}
```

```
for ( int i = expresión ; Condición ; i = expresión ) {
   ...
}
```

incorporación de los operadores postfijos ++ y --