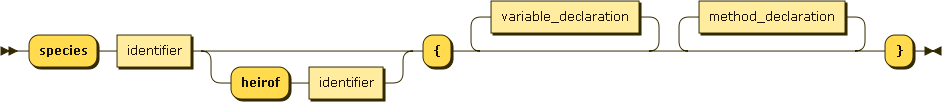
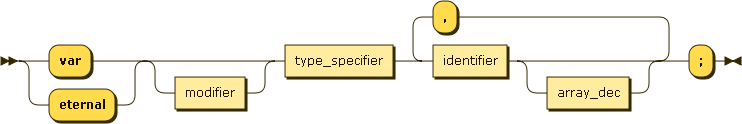
Nueva Clase:



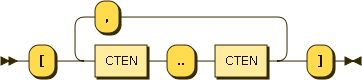
1. 2
2. Se checa si la clase no existe previamente, sino se da de alta en el directorio de clases. La variable actualSpecies = identifier y actualMethod = “species” porque se empieza en el ambiente de clase (global).
3. Checa que dicho padre exista, si es correcto se pone un apuntador a la clase padre en la llave “father” y se copia el tamaño de la clase padre en la clase actual.

Nueva variable:

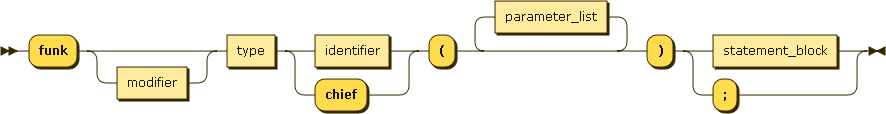
1 2 3 4

1. isVariable = true/false
2. actualModifier = true/false
3. actualType = type\_specifier
4. Dar de alta la variable en el directorio de procedimientos según sea el actualSpecies como atributo de clase o variable de función. Guardar el tipo, el modifier y si es variable de las variables anteriores. Checar si el tipo de la variable es valido: si no es de tipo primitivo entonces checar en la tabla de variable si la clase esta definida. Si es de tipo objeto, se parte la variable según todos los atributos de la clase y de las clases padre. Se saca una dirección y se guarda.

Declaracion de arreglo:



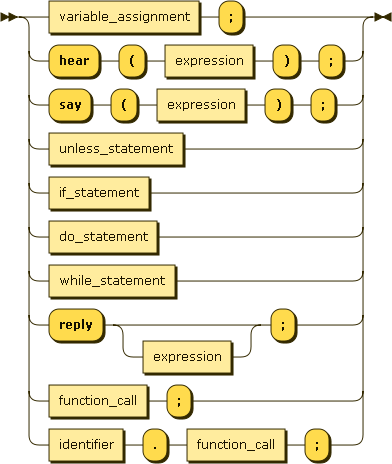
1. 2 3
2. Se checa que el tipo del objeto actual (actualType) es primitivo porque no se soportan arreglos de objetos. Se empieza a crear la “lista” de dimensiones. Se indica que es una variable dimensionada en el directorio de clases
3. Se hace un hash donde se guardan los limites superior e inferior, así como la “r”: (limite sup – lim inf + 1) \* r. Éste hash se agrega a la lista de dimensiones.
4. Con la ultima “r” se hace el cálculo de las “m” y las “k”. Se separa el espacio de memoria completo que requiere la variable.

Declaración de un método:

1 2 3 4 5 6

1. actualModifier = true (porque las funciones son por default públicas).
2. ActualModifier = modifier
3. actualType = type
4. Se da de alta la función en la tabla de clases/procedimientos, se valida que no exista otra función con el mismo nombre en la clase actual y en caso de que la función sea chief, se llena el cuádruplo de goto inicial para que sepa donde empezar el programa. Se checa que el tipo de la variable sea un tipo primitivo y se dan de alta los datos de modifier, type y el cuádruplo donde va a empezar la función. Se reinician los contadores locales.
5. Por cada tipo y id de parámetro: se checa que el tipo de parámetro sea valido, se da de alta el nombre en la tabla de variables de la función y se checa que no existe previamente una variable con el mismo nombre. Se saca la dirección del dato y se agrega junto con el tipo a la lista de argumentos de la función.
6. Se genera el cuádruplo de retorno de la función.

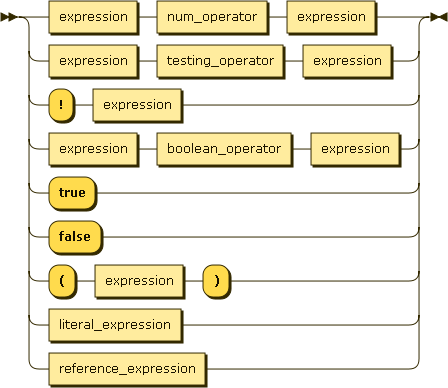
Expresiones:



1

1. Se genera cuádruplo con el valor de la expresión

Expresion:



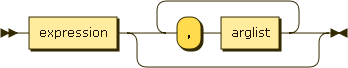
1

\*Sintácticamente al reducir la regla ya se tienen los valores y sus tipos, por lo que no hay una pila de operadores y/u operandos.

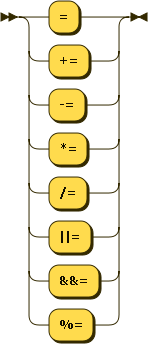
1. Se checa el tipo de ambas expresiones y se checa en el cubo que sea una operación válida. Se genera el cuádruplo con las dos direcciones de memoria

Resto de los diagramas:

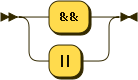
Arglist:



Assign operator:



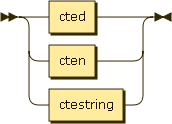
Boolean operator:



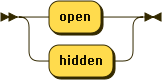
Do statement:

../../../../Downloads/diagram-6/diagram/do_statement.png

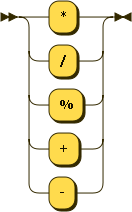
If statement:



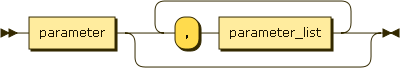
Modifier:



Num operator:



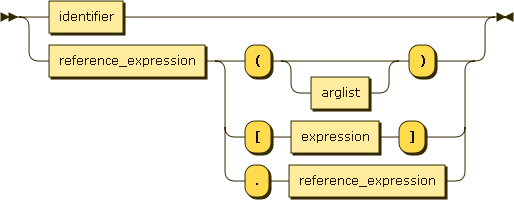
Parameter list:



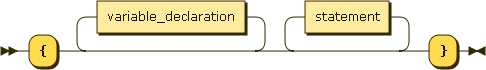
Parameter:

../../../../Downloads/diagram-6/diagram/parameter.png

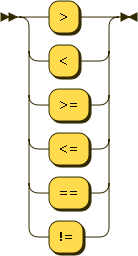
Reference expression:



Statement block:



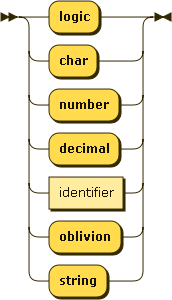
Testign operator



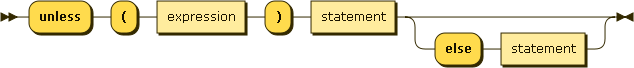
Type declaration:

../../../../Downloads/diagram-6/diagram/type_declaration.png

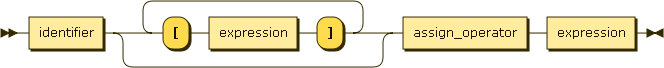
Type specifier:



Unless statement:



Variable Assignment:



While statement:

../../../../Downloads/diagram-6/diagram/while_statement.png