Desarrollo de aplicaciones avanzadas de ciencias computacionales

Actividad Final Mini Proyecto Parte 2

Presentado por:

Felipe Gabriel Yépez Villacreses

A01658002

Profesores:

Elda Guadalupe Quiroga González

Iván Mauricio Amaya Contreras

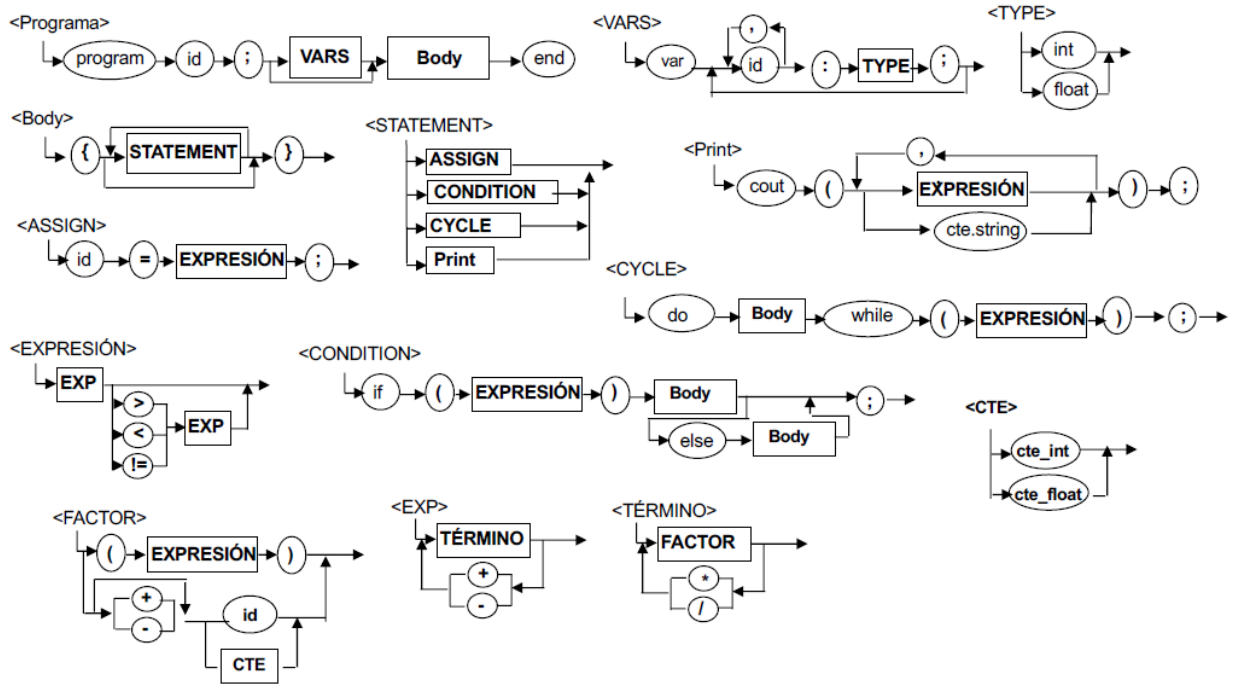
Edgar Covantes Osuna

Alexis Edmundo Gallegos Acosta

Fecha de entrega

Mi[ercoles 20 de mayo 2023

Mini Proyecto: Patito



Cubo Semántico

Para almacenar las consideraciones semánticas del lenguaje Patito es necesario implementar una tabla de consideraciones semánticas. Dentro de la misma se almacenan las operaciones válidas entre los diferentes tipos de dato que existen. Esta tabla se utilizará para poder validar las operaciones que se podrán y las que no se podrán dependiendo de los tipos de dato que se utilicen. Será posible identificar errores semánticos.

left_operator	right_operator	+	-	*	/	>	<	!=
int	int	int	int	int	float	bool	bool	bool
int	float	float	float	float	float	bool	bool	bool
float	int	float	float	float	float	bool	bool	bool
float	float	float	float	float	float	bool	bool	Bool

En la anterior tabla se plasmó el tipo de dato esperado al realizar las distintas operaciones existentes en Patito para cada combinación de tipos de dato posible.

Tabla de variables

Para la tabla de variables se escogió utilizar un Diccionario de Python dado que es eficiente acceder a los elementos almacenados. El acceso a los elementos se realiza con complejidad constante O(1) dado que su implementación está basada en tablas de hash. Al tener que almacenar variables con su respectivo tipo de dato es ideal el uso de un diccionario para utilizar la variable como llave y el tipo de dato como el valor almacenado. De igual forma es posible agregar elementos o modificar los existentes para lograr crear la tabla de variables del programa.

Para verificar que no se generen variables duplicadas es necesario buscar si una variable se dio de alta en la tabla anteriormente. Mediante el uso de un diccionario como estructura de datos, se puede realizar esta búsqueda en tiempo contante ya que se busca el elemento que puede o no puede existir usando hash.

Ejemplo de tabla de variables para la siguiente declaración de variables:

```
var num, i, j: int;
mean, num: float;
```

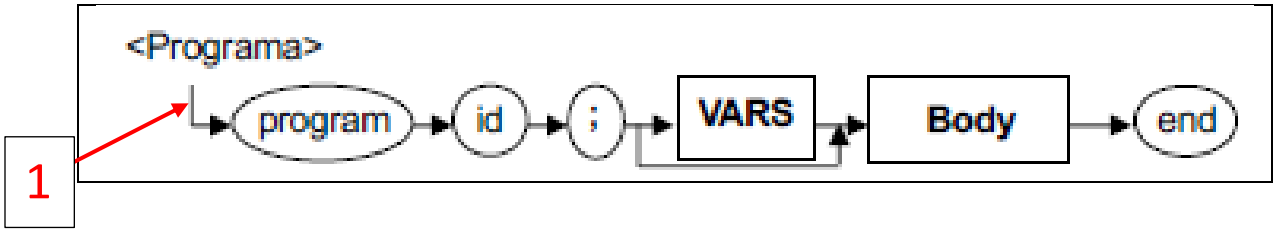
Llave (key)	Value (valor)
num	int
i	int
j	int
mean	float

Como se puede observar, en el código se declararon 5 variables y en la tabla de variables tan solo hay 4 dado que el código es capaz de detectar que una variable ya ha sido declarada anteriormente omitiendo su creación en la tabla de variables.

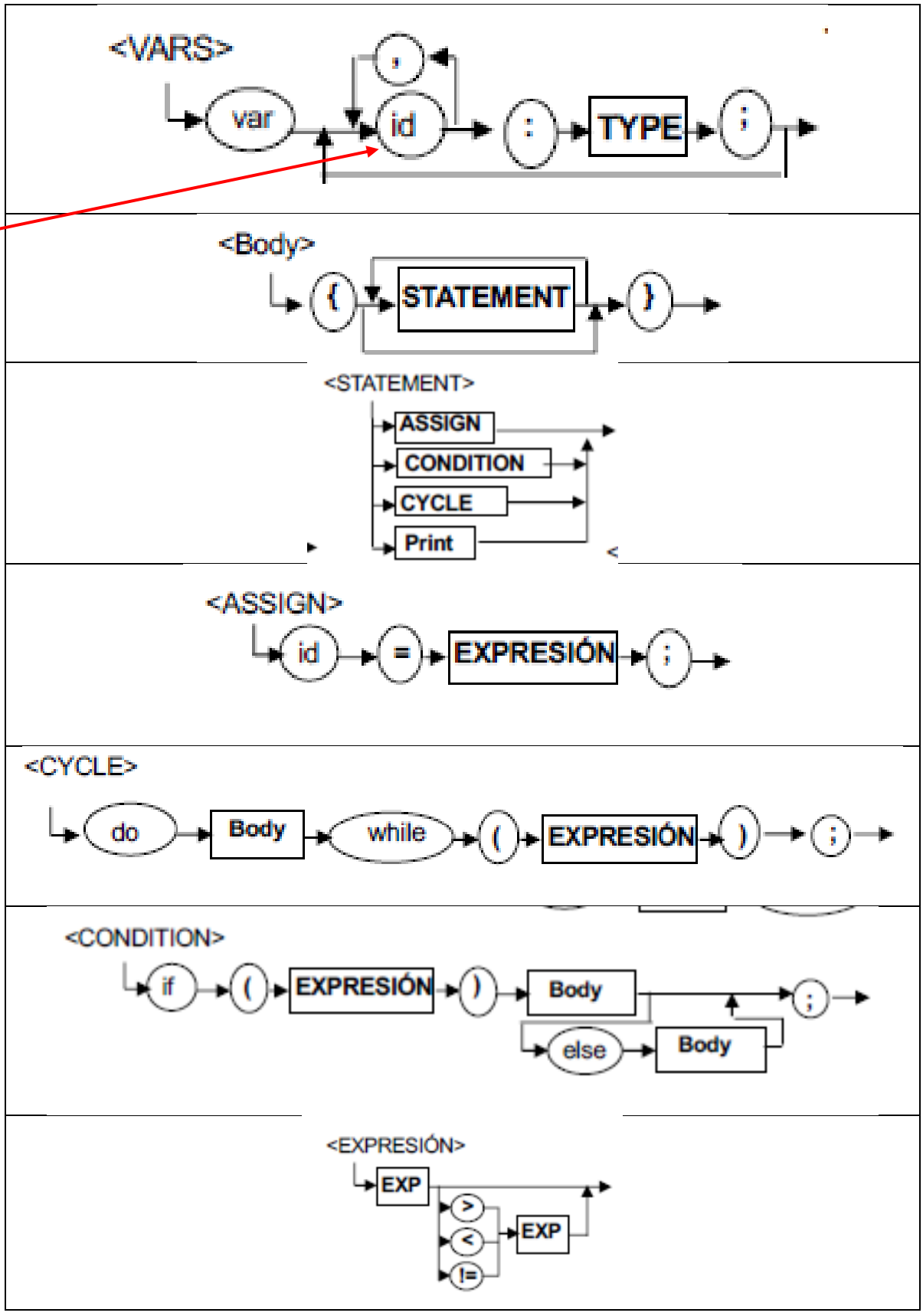
Los métodos de acceso principales de un diccionario son los siguientes:

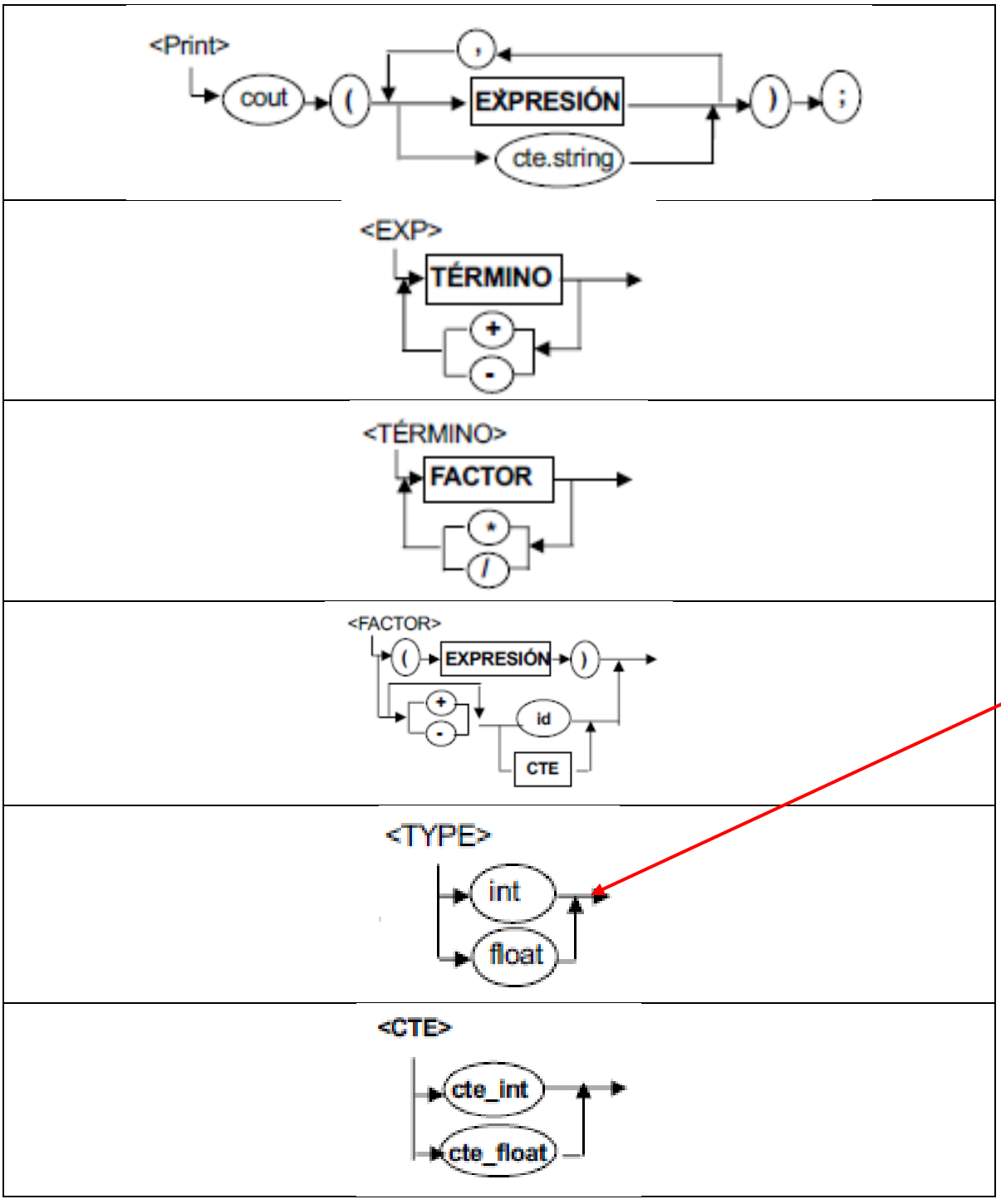
- diccionario[key] – devuelve valor del key
- diccionario.get(key) – valor del key
- diccionario.keys() – lista de llaves existentes en diccionario
- diccionario.values() – lista de los valores dentro del diccionario

Puntos Neurálgicos



2


<VARS>
var
id
,
:
TYPE
;
<Body>
{
STATEMENT
}
<STATEMENT>
ASSIGN
CONDITION
CYCLE
Print
<ASSIGN>
id
=
EXPRESIÓN
;
<CYCLE>
do
Body
while
(
EXPRESIÓN
)
;
<CONDITION>
if
(
EXPRESIÓN
)
Body
else
Body
;
<EXPRESIÓN>
EXP
>
<
!=
EXP



1. Crear tabla de variables
2. Buscar var_id en tabla de variables
   No se encuentra -> Agregar variable con tipo de dato None en tabla de variables
   Se encuentra -> Error, declaración de variable duplicada
3. Buscar en tabla de variables los registros con valores None
   Valor es None -> asignar val_type

Valor diferente a None -> ignorar

Casos de Prueba

Prueba Correcta

Se definen 4 variables de las cuales 3 son del tipo int y 1 es del tipo float.

```
var num, i, j: int;
mean: float;
```

Al probar con el caso de prueba anterior se obtiene impresa la tabla de variables.

```
Testing valid parser file...
          Type
Variable
num        int
i          int
j          int
mean     float
```

Prueba incorrecta

Se duplica la instancia de la variable num. La primera es del tipo int y la segunda es del tipo float.

```
var num, i, j: int;
mean, num: float;
```

Al ejecutar el caso de prueba, se muestra un mensaje de error indicando que la variable num ya existe anteriormente.

```
Testing invalid parser file...
Error in line 10 : Variable { num } already exists.
```