TEC | Tecnológico de Costa Rica

Campus Tecnológico Central Cartago

Compiladores e Intérpretes, Grupo 1

Etapa 0: Definición del Lenguaje

Prof. Kirstein Gätjens Soto

Daniel Sequeira Retana - 2019039641

Fecha de entrega: 23 de febrero

I Semestre
2024

Índice

Propuesta: Concurso de Nombres

Nombre propuesto: Táfac

Significado: Diablo celestial de los que tienen como función dar a los condenados su dosis diaria de tormento, que consiste en que los mastican, se los tragan y luego los expulsan, convertidos en bagazo, al defecar, orinar o **vomitar Logotipo propuesto:**

Extensión para los archivos propuesta: .taf

Definición del lenguaje

Estructura o parte del lenguaje	Elemento en el lenguaje	Traducciones del Malecu	Comentarios / Aclaraciones
Estructura del título del programa FC	nhequéquequé	Sonido onomatopéyico del chirrido de una puerta.	
Apertura sección constantes	octará	Traducción de piedra	escrito en piedra
Apertura sección de tipos	aíca	Traducción de varios	tipos equivale a que hay variedad
Apertura de la sección de variables	tí	Traducción de agua, en contraste a piedra	
Apertura de la sección de rutinas	urúma	Traducción de costumbre	
Apertura de la sección de prototipos	chóje	Traducción de proteger, un prototipo protege de un error declarando la función antes de su definición	
Punto de entrada del programa	cojcoj	Tocar la puerta para poder ingresar	
Sistema de asignación de constantes	caralh	Sonido de una piedra chocando con otra	
Sistema de asignación de tipos	putú	Traducción de cielo	Hay muchos tipos de cielo
Sistema de declaración de variables	sulí	Traducción de Cambio	
Tipo de dato entero	chátené	Entero o completo	
Tipo de dato carácter	filhílhi	Representación	
Tipo de dato string	purucpuruc	Cuerda	
Tipo de dato booleano	cochója	Doblado	
Tipo de dato conjunto	apúchá	Conjunto	
Tipo de dato archivo de texto	puríri	Contener	
Tipo de datos números flotantes	jané	Flotar	
Tipo de dato arreglos	cóye	Guardado/ Tener guardado	Los arreglos son sin límite de dimensión y son homogéneos y de cualquier tipo del lenguaje
Tipo de dato registros	lhará	Seña	

Tipo de datos atómico creativo	Número imaginario: úji	Traducción de imaginarse	
Literales booleanas	True: suírra / False: epéme	Traducción literal de verdadero y parcial de falso	
Literales de conjuntos	{ **'X1'**, **'X2'**,....,**'Xn'** }		Usa las literales de caracteres, separados por comas

Literales de archivos	{@ "c:\akw\tarea1.txt" , L @}		La primer literal de string es la ruta y nombre con extensión del archivo físico. La letra indica si es de lectura (L), escritura (S), ambas (D)
Literales de números flotantes	1.223		
Literales de enteros	-123 0xF4EC		Las de C, exactamente las de C.
Literales de caracteres	Con comilla simple: 'A' '\n' '\666'		Las de C, exactamente las de C.
Literales de strings	"Hola \n"		Las de C, exactamente las de C.
Literales del tipo de datos atómico creativo	3i un numero entero finalizando con una i		
Literales de arreglos	<[1,2,3,4]>		Llaves con corchetes
Literales de registros	úpalá [[ ]]	Traducción de casa	Brackets dobles
Sistema de acceso arreglos	<[]>		
Sistema de acceso strings	¿?		
Sistema de acceso registros	-->		
Asignación y Familia	óri / **<operación>**óri	Dar órdenes a alguien	Escoger un símbolo o palabra para la asignación y una sintaxis para las asignaciones combinadas.
Operaciones aritméticas básicas de enteros	+ - * % /		Suma Resta Multiplicación Residuo y División Entera
Incremento y Decremento	fay+ / tij-	Aumentar: fayátaqué / Disminuir: tíjye	Solo en prefijo
2 operaciones de enteros creativas	curíjurí para logaritmo / caquí: suma del piso x en el triángulo de pascal	Traducción de ritual / traducción de pilar	
Operaciones básicas sobre caracteres	Pasar a mayúsculas: taq / Pasar a minúsculas: jen / is alpha: mal / is digit: jat	Subir: taqué / Bajar: jené / Letra: malhióca / Numeroso: jatájatá	Pasar a mayúsculas, a minúsculas, is alpha, is digit, todas son unarias y en prefijo.
Operaciones lógicas solicitadas	and: muri / or: cóqui / xor: tiátiá / not: emé	Traducción de and y Traducción de suceder (un or es esperar que algo suceda). Xor: Traducción de exactamente Not: Traducción de no	

Operaciones de Strings solicitadas	Concatenar: catáqui / Length: lharátesufá / cortar: joné / recortar: nhalh / encontrar: cochéqui	Adherir: catáqui / Cosa larga: lharátesufá / Cortar: joné / Ideófono del sonido del cortar de un solo golpe un objeto: nhalh / Encontrar: cochéqui	Concatenar length cortar recortar find
Operaciones de conjuntos solicitadas	Agregar: erronh / Borrar: quené / Union: fuinhye / Interseccion: sulí / PerteneceA: pcalúfa / Vacío?: culíte	Añadir: erronh / Borrar: quené / Mezclar: fuinhye / Cruzar: sulí / Miembro: pcalúfa / Ser vaciado: culíte	Agregar Borrar Union Interseccion PerteneceA Vacío?
Operaciones de archivos solicitadas	Abrir: quijí / Cerrar: eloc / Escribir: lhioc / Leer: lhaic / Crear: quijérri / Asociar: láte	Abrir: quijí / Cerrar: eloc / Escribir: lhioc / Leer: lhaic / Crear: quijérri / Relación: láte	abrir cerrar escribir leer crear asociar
Operaciones de números flotantes	Suma: erroc / Resta: sírru / Multiplicación-> fují / División: palé	Añadir: erroc / Quitar: sírru / Acumulación: fují / Dividir: palé	suma, resta, multiplicación, división (binarias)
Operaciones del tipo de datos	i+ i* i/ i-		Suma, multiplicación, división, resta

atómico creativo			
Operaciones de comparación solicitadas	> < = >= <= ><		Se comparan enteros y retornan booleanos
Manejo de Bloques de más de una instrucción	CojCoj		Hay que pensar el de cierre. O sugerir un par diferentes si encuentran algo más divertido.
Instrucción while	pují <exp> sóta / pují (<exp>)	Dedicar el tiempo a algo: pují / Hacer: sóta	while <exp> do while (<exp>)
Instrucción if-then-else	if: arú / else: ótacá		Como el ifTrue de Smalltalk!! then y else opcionales y en cualquier orden.
Instrucción switch	Switch: lhifíji / case: palhá / default: quírra	Cambiar: lhifíji / Indica que llego el momento de hacer algo: palhá / Origen: quírra	switch case default Como el de C.
Instrucción Repeat-until	paípaí / yája	Repetidamente: paípaí / Hasta: yája	como el repeat until de pascal.
Instrucción For	ajá	Para: ajá	como el for de pascal no de C puede ser como el de python
Instrucción With	tiní	Con: tiní	como el whith de pascal
2 Instrucciones creativas	Tarráni: hacer una pausa en la ejecución hasta que se reciba una entrada en el teclado / Chequéle: equivalente a un cls, limpiar consola	Traducción de tiempo	
Instrucción break	foróforóye	Romperle la entrada de la cueva a una taltuza	como el de C
Instrucción continue	taráne	Seguir la línea de algo como se determinó previamente	como el de C
Instrucción Halt	lauc	Terminar	Detiene la ejecución del programa completamente.
Encabezado de funciones	lufá **id** (**...**) : **tipo**	Actividad	buscar palabra melecu

Encabezado de procedimientos	ónhaónha **id** (**...**)	Conducta	buscar palabra melecu
Manejo de parámetros formales	x(.a, .b, .c)		Deben ser consistentes con la declaración de variables. Hay que pensar la forma de decir que un parámetro va por referencia y no por valor.
Manejo de parámetros reales	x(a, b, c)		
Instrucción return	jalac	Devolver	Hay dos trabajos diferentes que hace el return de C. ¿Los separamos? **Sí, proponemos que jalac no cumpla la función de terminar el programa**
Operación de size of	níca	De este tamaño	tamaño ya sea de un tipo o de una expresión.
Sistema de coerción de tipos	ü(nombre_del_tipo)		**expresión (pensar un operador en malecu) tipo**
Manejo de la entrada estándar	tioc**tipo**	Entrar: tioc	Los seis tipos atómicos
Manejo de la salida estándar	píte**tipo**	Salir: píte	Los seis tipos atómicos
Terminador	;		Es el punto y coma y es un terminador

Índice de pruebas

Asignaciones.txt

Contiene ejemplos de las asignaciones

instrucciones.txt

Contiene ejemplos de las instrucciones

Operaciones.txt

Contiene ejemplos de las instrucciones

Tipos-de-datos.txt

Contiene ejemplos de los tipos de datos

Propuesta de Gramática de Táfac

- Estructura del título del programa **nhequéquequé**

En código:

```
## esto va al inicio del programa
## y declara el nombre del mismo
nhequéquequé miPrograma;
```

<apertura> ::= nhequéquequé id ;

- Apertura de sección de constantes **octará**

En código:

```
##apertura de sección de constantes: octará
octará

    ## arrá instrucción prefija que declara x1
    arrá x1 45;
    arrá x2 'd';
    arrá x3 maíca; ##maica es False
```

<const> ::= octará <expresión>

<expresión> ::= ... <expresión>

<expresión> ::= ε

- Apertura de sección de tipos **aíca**

En código:

```
##apertura de sección de tipos: aíca
aíca

    ## putú es una instrucción prefija
    putú Edad chátené;
```

<tipoSec> ::= aíca <expresión>

- Apertura de sección de variables **tilhtilh**

En código:

```
##apertura de sección de variables: tilhtilh
tilhtilh

    ##chátené es tipo entero y nicó su asignación,
    chátené x4 nicó 23;
    coyé chátené x5 ta 10;              ## esto de
    ílo x6 nicó "Esto es un string";    ## ílo es
```

<variableSec> ::= tilhtilh <expresión>

- Apertura de sección de rutinas **purú**

En código:

```
## sección de rutinas (funciones y procedimientos): purú
purú

macorróca afá esUnaVocal (malhióca Letra)
cojcoj                  ## se puede interpretar como {
```

<rutinasSec> ::= purú <expresión>

- Apertura de sección de prototipos **chá**

En código:

```
## sección de prototipos (se refiere a la declar
chá
    ##macorróca es el encabezado de la función,
    macorróca chátené fibonacci (chátené N);
    óri factorial (chátené N, suí chátené Resp);
    macorróca afá esUnaVocal (malhióca Letra);
```

<prototiposSec> ::= chá <expresion>

- Punto de entrada del programa **cojcoj**

En código:

```
cojcoj
    châtené contador nicó 10;
    châtené aumentador nicó 0;

    ##ejemplo de un while
    ótacá(contador >= 0) sóta
```

<coj> ::= cojcoj <expresion>

- Sistema de asignación de constantes **arrá**

En código:

```
## arrá instrucción prefija que
arrá x1 45;
arrá x2 'd';
arrá x3 maíca; ##maica es False
```

<asigConst> ::= arrá id <literalX> ;

- Sistema de asignación de tipos **putú**

En código:

```
## putú es una instr
putú Edad châtené;
```

<tiposConst> ::= putú id <tipoDato> ;

- Sistema de asignación de variables **nicó**

En código:

```
##châtené es tipo entero y nicó s
châtené x4 nicó 23;
coyé châtené x5 ta 10;
ílo x6 nicó "Esto es un string";
afá x7 nicó tócu;
```

<> ::= <tiposDeDatos> id nicó <literalX> ;

- Tipos de datos

En código:

```
chátené campo1;
malhióca campo2;
ílo campo3;
afá campo4;
```

<tiposDeDatos> ::= <tipoDato> id <asignacionX> ;

<asignacionX> ::= ε

<tipoDato> ::= chátené

<tipoDato> ::= malhióca

<tipoDato> ::= ílo

<tipoDato> ::= afá

<tipoDato> ::= apúchá

<tipoDato> ::= lhalámi

<tipoDato> ::= jané

<tipoDato> ::= coyé <tipoDato> id ta

- Tipo de dato booleano **afá**

En código:

```
afá x7 nicó tócu;
```

<booleanoTipo> ::= afá id <asignacionX> ;

<asignacionX> ::= nicó <booleano>

<asignacionX> ::= ε

<booleano> ::= tócu

<booleano> ::= maíca

- Tipo de dato registro: **porétecá ... turé**

En código:

```
##Esto es el equivalente
porétecá
    chátené campo1;
    malhióca campo2;
    ílo campo3;
    afá campo4;
turé X nicó .... ;
```

<reg> ::= porétecá <listaDatos> turé id <asignacionX> ;

<listaDatos> ::= <nuevoDato>

<listaDatos> ::= <nuevoDato> <listaDatos>

<asignacionX> ::= ε

● Literales de conjuntos

En código:

```
apúchá x8 nicó {x4,x5,x6,x7};
```

<S> ::= apuchá id <asignacionX> ;

<asignacionX> ::= nicó { <conjunto> } | ε

<conjunto> ::= id

<conjunto> ::= id , <conjunto>

● Literales de archivos

En código:

```
lhalámi x9 nicó {{ "c:\akw\tarea1.txt" , L }} ;
```

<S> ::= lhalámi id <asignacionX> ;

<asignacionX> ::= nicó {{ <stringRuta> , <accionArchivo> }} | ε

<accionArchivo> ::= L

<accionArchivo> ::= S

<accionArchivo> ::= D

● Literales de números flotantes

En código:

```
jané x10 nicó 1.89;
```

Es importante resaltar que los literales de los flotantes en Táfac son iguales a los del lenguaje C, por lo que la gramática de C influenció fuertemente nuestra propuesta.

<S> ::= jané id <asignacionX> ;

<asignacionX> ::= nicó <flotante> | ε

<flotante> ::= <entero> . <parte_fraccionaria> <exponente>

<exponente> ::= E + <entero>

<exponente> ::= E - <entero>

<exponente> ::= E <entero>

<exponente> ::= ε

<entero> ::= <digito>

<entero> ::= <digito> <entero>

<entero> ::= ε

<parte_fraccionaria> ::= <digito>

<parte_fraccionaria> ::= <digito> <parte_fraccionaria>

<digito> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

● Literales de enteros

En código:

```
chátené x4 nicó 23;
```

Es importante resaltar que los literales de los enteros en Táfac son iguales a los del lenguaje C, por lo que la gramática de C influenció fuertemente nuestra propuesta.

<S> ::= chátené id <asignacionX> ;

<asignacionX> ::= nicó <entero> | ε

<entero> ::= <entero_decimal>

<entero> ::= <entero_octal>

<entero> ::= <entero_hexadecimal>

<entero_decimal> ::= <digito_no_cero>

<entero_decimal> ::= <digito_no_cero> <digitos_decimales>

<entero_decimal> ::= 0

<entero_octal> ::= 0

<entero_octal> ::= 0 <digitos_octales>

<entero_hexadecimal> ::= 0 X <digitos_hexadecimales>

<digitos_decimales> ::= <digito_decimal>

<digitos_decimales> ::= <digito_decimal> <digitos_decimales>

<digitos_octales> ::= <digito_octal>

<digitos_octales> ::= <digito_octal> <digitos_octales>

<digitos_hexadecimales> ::= <digito_hexadecimal>

<digitos_hexadecimales> ::= <digito_hexadecimal> <digitos_hexadecimales>

<digito_no_cero> ::= 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<digito_decimal> ::= 0 | <digito_no_cero>

<digito_octal> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7

<digito_hexadecimal> ::= <digito_decimal> | A | B | C | D | E | F

● Literales de caracteres

En código:

```
malhióca x11 nicó ':';
```

Es importante resaltar que los literales de los caracteres en Táfac son iguales a los del lenguaje C, por lo que la gramática de C influenció fuertemente nuestra propuesta. Se evita escribir todos los posibles caracteres por comodidad visual, pero se entiende que son los mismos que se aceptarían en C.

<S> ::= malhióca id <asignacionX> ;

<asignacionX> ::= nicó <caracter> | ε

<caracter> ::= ‘ <caracter_literal> ‘

<caracter> ::= ‘ <escape_secuencia> ‘

<caracter_literal> ::= cualquier carácter visible excepto ' y \

<escape_secuencia> ::= \ <combinacionesValidasC>

● Literales de strings

En código:

```
ílo x6 nicó "Esto es un string";
```

Es importante resaltar que los literales de los strings en Táfac son iguales a los del lenguaje C, por lo que la gramática de C influenció fuertemente nuestra propuesta. Se evita escribir todos los posibles caracteres por comodidad visual, pero se entiende que son los mismos que se aceptarían en C.

<S> ::= ílo id <asignacionX> ;

<asignacionX> ::= nicó <string> | ε

<string> ::= “ <string_literal> <escape_secuencia> ”

<string_literal> ::= <combinacionesStringsC>

<string_literal> ::= ε

<escape_secuencia> ::= \ <combinacionesValidasC>

<escape_secuencia> ::= ε

- Literales de arreglos

En código:

```
coyé châtené x12 nicó {[ 1,2,3,4 ]};
```

<S> ::= coyé <tipoDato> id <asignacionX> ;

<asignacionX> ::= nicó <arreglo> | ε

<arreglo> ::= { [<datosDeTipoDato>] }

<datosDeTipoDato> ::= <dato>

<datosDeTipoDato> ::= <dato> , <datosDeTipoDato>

<dato> ::= id

<dato> ::= <datoDeTipoDato>

-

En código:

Literales de registros

```
porétecá                                ## con
    chátené campo1;                     ## Los
    malhióca campo2;                    ## lue
    ílo campo3;
    afá campo4;
turé X nicó {@ 1, 'a', "string", x7 @} ;
```

<S> ::= porétecá <listaDatos> turé id <asignacionX> ;

<asignacionX> ::= nicó <registro> | ε

<registro> ::= { @ <datosCorrespondientes> @ }

<datosCorrespondientes> ::= <dato>

<datosCorrespondientes> ::= <dato> , <datosDeTipoDato>

<dato> ::= id

<dato> ::= <datoCorrespondiente>

● Sistema de acceso de arreglos

En código:

```
x5[1] := 6;
```

<S> ::= id [<accesador>] := <dato> ;

<accesador> ::= id

<accesador> ::= <entero>

● Sistema de acceso de strings

En código:

```
x6 $$ 3;
```

<S> ::= id $$ <entero> ;

● Sistema de acceso de registros

En código:

-

En código:

```
campo1@A := 25;
```

<S> ::= idDato @ idRegistro := <datoTipoDato> ;

Asignación: **:= += -= *= %= /=**

```
chátené x4 nicó 23;
campo1@A := 25;

x4 += 2;
x4 -= 1;
x5 *= 10;
x6 %= 2;
x7 /= 3;
```

<asiganciones> ::= idDato <asignacion> <datoTipoDato> ;

<asignacion> ::= :=

<asignacion> ::= +=

<asignacion> ::= -=

<asignacion> ::= *=

<asignacion> ::= %=

<asignacion> ::= /=

● Operaciones aritméticas básicas de enteros: **+ - * % /**

En código:

```
chátené x4 nicó 23;
chátené x5;

x5 := x4 + 5;
x5 := x4 - 3;
x5 := x4 * 4;
x5 := x4 / 2;
x5 := x4 % 6;
```

<S> ::= <dato> <operacionEntero> <dato> ;

<dato> ::= id

<dato> ::= <entero>

-

En código:
<operacionEntero> ::= *

<operacionEntero> ::= /

<operacionEntero> ::= %

<operacionEntero> ::= +

<operacionEntero> ::= -

Decremento/incremento de variables: **jené y taqué**

```
## Declara una variable de tipo
## entero con asignación 23 e incremento
chatené x nicó 23;
x jené;
```

<DecIncremento> ::= id <cremento>

<cremento> jené

<cremento> taqué

● Operaciones básicas sobre caracteres: **>> << &? #?**

En código:

-

En código:

<operacionesCaracteres> ::= <operadorCaracter> id

<operadorCaracter> ::= >>

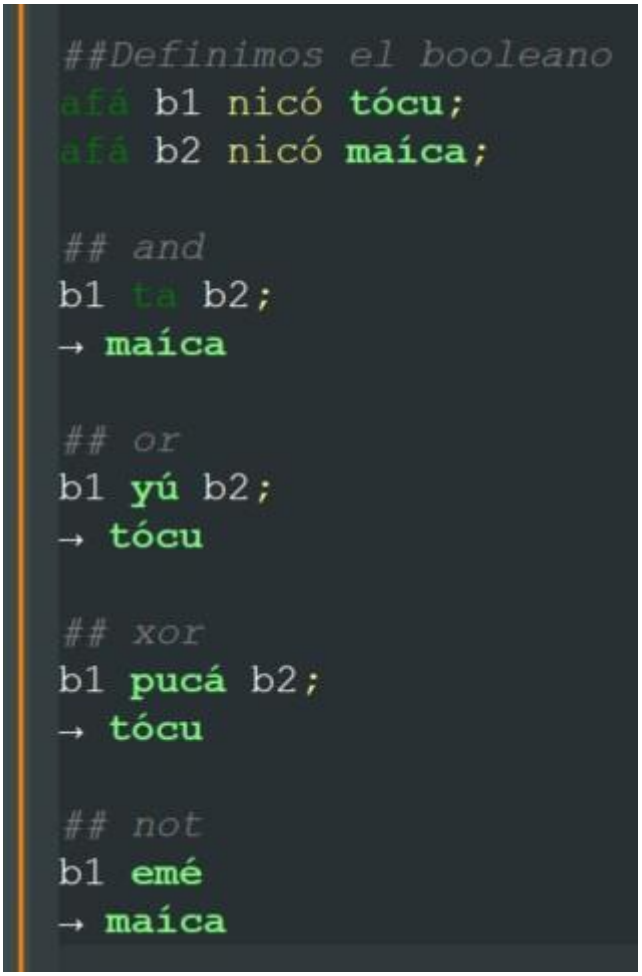
<operadorCaracter> ::= <<

<operadorCaracter> ::= &?

<operadorCaracter> ::= #?

- Operaciones lógicas: **ta yú pucá emé**

En código:



<operacionesLogicas> ::= id <operadorBooleano> id

<operadorBooleano> ::= ta

<operadorBooleano> ::= yú

<operadorBooleano> ::= pucá

<operadorBooleano> ::= emé

- Operaciones con strings: **Concatenar: catáqui / Length:lharátesufá / cortar: joné / recortar: nhalh / encontrar:cochéqui**

En código:

```
##Definir las variables
ilo string1 nicó "Esto es un string";
ilo string2 nicó "el veloz murciélago";

##concatenar
string1 catáqui string2
→ "Esto es un stringel veloz murciélago"

## lenght
lharátesufá string1
→ 17

## cortar
joné[0:5] string1
→ "Esto e"

##recortar
nhalh[0:5] string2
→ "oz murciélago"

##encontrar
cochéqui["un"] string1
→ 8
```

<operacionesStrings> ::= id catáqui id

<operacionesStrings> ::= lharátesufá id

<operacionesStrings> ::= <operadorManejoString> [<literalEntera> : <literalEntera>] id

<operadorManejoString> ::= joné

<operadorManejoString> ::= nhalh

<S> ::= cochéqui [<literalString>] id

- Operaciones con conjuntos: **Agregar: erronh / Borrar:quené / Union: fuinhye / Interseccion: sulí / PerteneceA: pcalúfa / Vacío?: culíte**

En código:

```
## agregar
erronh con1 'x5';
→ {'a1','x1','x2','x3','x4','x5'}

## borrar
quené con1 'x1';
→ {'a1','x2','x3','x4'}

## Union
fuinhye con1 con2
→ {'a1','x1','x2','x3','x4','a1','y1','y2','y3','y4'}

## intersección
sulí con1 con2
→ {'a1'}

## pertenece
pcalúfa con1 'a1'
→ tócu

## vacio
culíte con1
→ maíca
```

<operacionesConjuntos> ::= <operadorConjunto> id <literalX>

<operadorConjunto> ::= erronh

<operadorConjunto> ::= quemé

<operadorConjunto> ::= pcalúfa

<operacionesConjuntos> ::= <operadoConjunto> id id

<operadorConjunto> ::= sulí

<operadorConjunto> ::= fuinhye

<operacionesConjuntos> ::= <operadorConjunto> id

<operadorConjunto> ::= culíte>

- Operaciones con archivos: **Abrir: quijí / Cerrar: eloc / Escribir: lhioc / Leer: lhaic / Crear: quijérri / Asociar: láte**

En código:

```
lhalámi cheto nicó {{ "c:\akw\tarea1.txt" , L }}

## abrir
quiji cheto;

##cerrar
eloc cheto;

## escribir
lhioc cheto "texto a escribir";

## crear
quijérri muffin;

## asociar
láte muffin cheto;
```

<manejoArchivo> ::= <operadorArchivo> id

<manejoArchivo> ::= <operadorArchivo> id id

<manejoArchivo> ::= <operadorArchivo> id <literalString>

<operadorArchivo> ::= quijí <operadorArchivo> ::= eloc

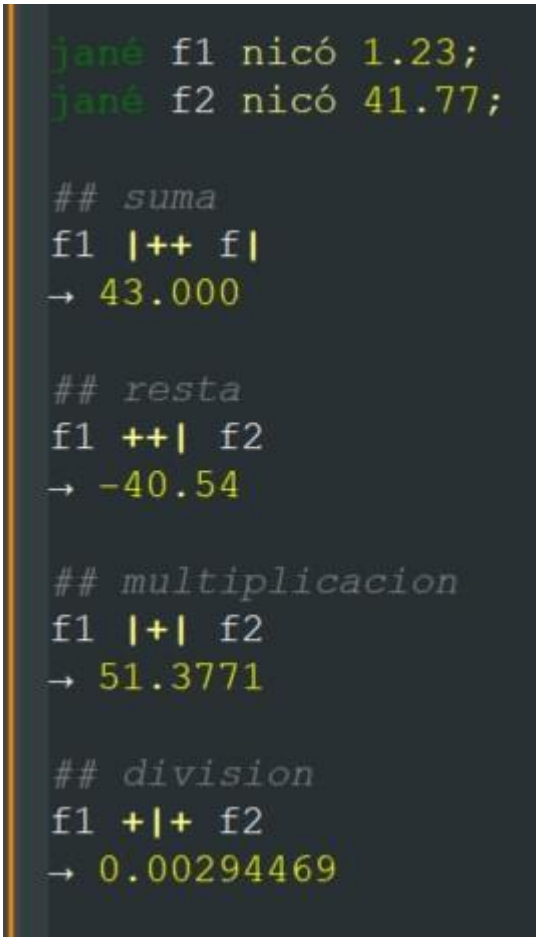
<operadorArchivo> ::= lhioc

<operadorArchivo> ::= quijérri

<operadorArchivo> ::= láte

- Operaciones con números flotantes:**|++ ++| |+| +|+**

En código:



<operacionesFlotantes> ::= id <operadorFlotante> id

<operacionesFlotantes> ::= <literalFlotante> <operadorFlotante> id

<operacionesFlotantes> ::= id <operadorFlotante> <literalFlotante>

<operadorFlotante> ::= |++

<operadorFlotante> ::= ++|

<operadorFlotante> ::= |+|

<operadorFlotante> ::= +|+

- Operaciones de comparación: **< > = >= <= ><**

En código:

<operaracionesComparacion> ::= id <operadorComparacion> id

<operaracionesComparacion> ::= id <operadorComparacion> <literalX>

<operaracionesComparacion> ::= <literalX> <operadorComparacion> id

<operadorComparacion> ::= <

<operadorComparacion> ::= >

<operadorComparacion> ::= =

<operadorComparacion> ::= >=

<operadorComparacion> ::= <=

<operadorComparacion> ::= ><

- Manejo de Bloques de más de una instrucción: **Cojcoj fuchí**

```
cojcoj
    pítechátené i;
    íca i;
fuchí
```

<bloqueInstrucciones> ::= cojcoj <instruccion> <instrucción> fuchí
<instrucción> :: = … <instrucción>
<instruccion> ::= …
<instrucción> :: = ε

- Instrucción while

```
##ejemplo de un while
ótacá contador >= 0 sóta ## while(contador >= 5) do
cojcoj
    contador jené;        ## esto es un contador--
    aumentador taqué;     ## esto es un aumentador++
    taráne                ## continue (no es obligatorio)
fuchí
```

<while> ::= ótacá <operaracionesComparacion> sotá <instrucción>

Instrucción if-then-else

```
##ejemplo de if-then-else
arú aumentador > 5 nocófa    ## if (aumentador > 5) then
cojcoj
    aumentador := 0;
    contador := 0;
fuchí
tiáfa                        ## else
cojcoj
    aumentador := 5;
    contador := 5;
fuchí
```

<ifThenElse> ::= arú <operaracionesComparacion> nocófa <instruccion> tiáfa <instruccion>

<ifThenElse> ::= arú <operaracionesComparacion> nocófa <instruccion>

- **Instrucción switch**

```
torré Letra      ## torré es switch
cojcoj
    ári 'a' :   ## ári es el case
    ári 'e' :
    ári 'i' :
    ári 'o' :
    ári 'u' :
        jalac tócu; ## jalac es return, o
        foróforóye; ## este es el break
    mailhíco:       ## este es el default
        jalac maíca;## return false
fuchí
```

<switch> :: = torré id cojcoj <ari> mailhíco <instruccion> fuchí

<ari> ::= ári <literal> : <instruccion> <ari>

<ari> ::= ε

Instrucción repeat-until

```
## Ejemplo de repeat-until
paípaí   ## do
    aumentador += 1;
yája aumentador < 10; ## while
```

<repeatUntil> ::= paípaí <instruccion> yája <operaracionesComparacion> ;

- **Instrucción for**

```
##ejemplo de un for
ajá chátené i := 5 Iha i >= 0 telele i jené sóta ## fo
cojcoj                ## el cojcoj al igual que el {} en
    pítechátené i;    ## píte es salida estandar o sea e
    íca i;            ## sizeof de i
fuchí
```

<for> ::= ajá <tiposDeDatos> <asiganciones> lha <operaracionesComparacion> telele id <DecIncremento> sóta <instruccion>

- **Instrucciones break, continue y hault**

En código:

```
chátené contador nicó 10;
chátené aumentador nicó 0;

ótacá(contador >= 0) sóta
cojcoj
    contador jené;
    aumentador taqué;
    arú (aumentador = 5) nocófa
        foróforóye
    taráne
fuchí

lauc
```

<break> ::= foróforóye

<continue> ::= taráne

<halt> ::= lauc

Encabezado de funciones

<encabezadoFun> ::= macorróca <tipo> id ( )

- **Encabezado de procedimientos**

<encabezadoProc> ::= óri id ( )

- **Manejo de parámetros formales**

<parametroFormal> ::=(<tiposDeDatos> id , <tiposDeDatos> id , sui <tiposDeDatos> id
)

- **Manejo de parámetros reales**

<parámetrosReales> ::=id (a , b , c)

- **Instrucción return**

<return> ::= jalac ;

- **Operación sizeof**

<sizeOf> ::= icá id ;

- **Sistema de coerción de tipos**

<coerción> ::= tirríque (<tiposDeDatos>) id ;

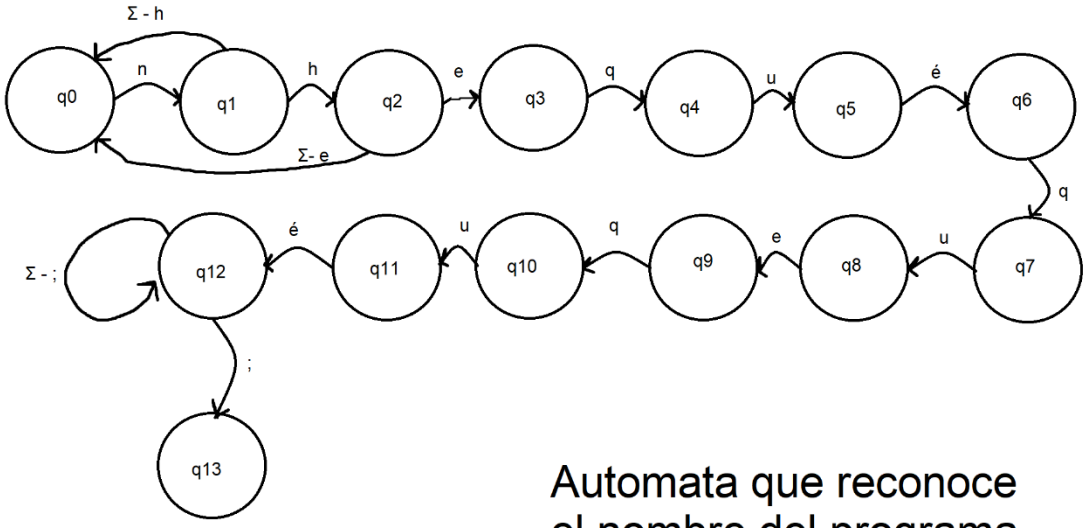
- **Salida Estándar**

<salidaES> ::= lac<tiposDeDatos> id

- **Entrada Estándar**

<entradaES> ::= píte<tiposDeDatos> id

Autómata



Automata que reconoce
el nombre del programa